Efficient pairing computation on Barreto-Naehrig elliptic curves of embedding degree 12

Florence Clerc, Rémi Géraud

7th May 2012

Contents

Chapter 1

Introduction

The history of cryptography dates back to Antiquity, where the ability to exchange messages in a secure way was linchpin to military or political success. The central idea of cryptography is to transform a message so that only certain individuals can access its contents. For everyone else, the encrypted message is intended to be meaningless and useless. Unlike steganography, in which one would try to hide the message in such a way that undesired readers wouldn't find it, a cryptographic scheme should retain its strength even under analysis.

Therefore, more and more complicated techniques were found, in order to distort and transform the original message, with the hope that cryptanalysts would fail to break the code.

Until recently however, all cryptographic schemes had a common weakness: in order to communicate information securely, a secret had to be shared. Of course, it is easier to share small secrets, small messages, than larger ones. Therefore many methods rely on a small piece of information, the *key*, that both parties agree upon before engaging in a more secure communication that uses cryptographic tools and the key to protect a larger piece of information.

Early methods are said to be *symmetric*, because generally the same key is used both to encrypt and decrypt the message. Why is that a weakness?

- If the key is found by someone else, the said person can decrypt all the messages that have been sent ;
- Similarly, this person can forge messages using the key ;
- Eventually, the study of encrypted messages might enable someone to find or guess the key.

Some workarounds are fairly straightforward: use complex keys and cyphers, change them often. But the problem remains: how to share a secret in the first place?

The modern answer to that question came with the advent of asymmetric cryptography in the 1970s, also known as public-key cryptography. Within such a framework, it is possible to build secure channels (Diffie-Hellman algorithm, see below), to check the identity of the sender of a message, etc. Asymmetric

cyptography is now at the heart of everyday applications (smart cards, electronic commerce...) as well as in civil and military activities.

Although a very powerful concept, it is not impervious to subtle or more brutal attacks, and cryptographic schemes have to be cleverly designed in order to avoid known pitfalls and shortcuts that would allow an attacker to break the code. Research in this domain is thus very active, both in designing new algorithms (cryptography) and in trying to weaken these algorithms by attacking them (cryptanalysis). A general framework to devise asymmetric cryptographic schemes is the theory of Elliptic Curve Cryptography (ECC).

Since its inception in the 1980s [20, 16], Elliptic Curve Cryptography (ECC) is thought to be one of the most secure and powerful cryptosystems[1].

ECC was first introduced because of its interesting group structure (see section 2), which could readily be used in conjunction with many existing methods. Since recent years however, research has shown a vivid interest in using the intrisic properties of elliptic curves. These additional properties were initially thought as weaknesses, but eventually, they enabled new protocols that were impossible before. Pairings (see chapter 5) are a vibrant example, and are becoming increasingly useful in elaborating new cryptosystems.

1.1 Organisation and aim of this document

In chapter 2, we recall basic mathematical definitions. In chapter 3, we briefly introduce elliptic curves. This allows to explain in chapter 4 how point multiplication can be improved for curves on which an efficient endomorphism is known. Considering the importance of cryptography, many methods to improve the computation have been developped. We detail one of them, the GLV method. Whereas in the previous chapters, the work is done using sums and scalar-multiplication, in chapter 5 we work on pairings. Based on previous works such as [13], we explain a method that allows to compute faster pairings. In chapter 6, we adapt this method to curves of degree 12.

1.2 Acknowledgements

The authors would like to express their most respectful gratitude to Damien Vergnaud for supervising this work, bringing new perspectives to our avid curiosities, and providing us with this extraordinary opportunity to delve into a bleeding-edge topic. We thank him for all the time he has spent helping us and for his patience in this task. He has allowed us to discover a field we had barely heard about and to apply algebra we had learnt in classes préparatoires in a different context. Thanks to his efforts it has been an enriching experience.

We also thank M. Lamoureux for his enthusiast support and trust, and for his communicative liking of Mathematics as a discipline.

[1] The National Security Agency of the United States of America recommends it in sensitive applications: http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml

Chapter 2

Mathematical background

In this chapter we recall basic definitions of algebraic structures. We are essentially interested in finite fields (also known as Galois fields).

2.1 General notations

When a and b are integers, we denote by $a \mid b$ that a divides b. Conversely, $a \nmid b$ means that a does not divide b. When a and b are congruent modulo n, we shall write $a \equiv b \pmod{n}$. The greatest common divisor of two numbers a and b is the largest number c such that $c \mid a$ and $c \mid b$, and is noted gcd(a, b).

2.2 Groups, rings and fields

Definition 1. *Let G a set such that $G \neq \emptyset$ and let $+$ a binary operation on G. $(G, +)$ is said to be a* group *if:*

- *$+$ is associative: $\forall x, y, z \in G,\ x + (y + z) = (x + y) + z$;*
- *G has an identity $0 \in G$: $\forall x \in G,\ 0 + x = x = x + 0$;*
- *every element $x \in G$ has an inverse element $y \in G$: $x + y = 0 = y + x$. The inverse of x is noted $-x$.*

If for any two $x, y \in G$, $x + y = y + x$, then G is said to be abelian *(or* commutative*) ; otherwise G is said to be* non-commutative.

Definition 2. *Given a group G noted additively, a* generating set *S not contained in any proper subgroup of G is such that :*

$$\forall x \in G, x = \sum_{y \in S} a_y.y$$

where a_y is an integer such that $a_y.y = \underbrace{y + ... + y}_{a_y \ times}$

An element of S is called a generator of G. *If G can be generated from only one element, then G is a* cyclic group.

As an example, $(\mathbb{Z}/n\mathbb{Z}, +)$, the numbers modulo n with the usual addition law, form a group. This group is commutative and cyclic. Any number d prime with n is a possible generator of G.

Definition 3. *A set R equipped with two binary operations $+ : R \times R \to R$ and $\cdot : R \times R \to R$ called* addition *and* multiplication *is a* ring *when:*

- $(R, +)$ *is an abelian group ;*
- $(R, \cdot)$ *is a monoid : multiplication is associative, has an identity* 1 *such that $1 \cdot a = a \cdot 1 = a$ holds, and for all $a, b \in R$, $a \cdot b \in R$.*
- *The operations are distributive :*

$$\forall a, b, c \in R, a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$
$$\forall a, b, c \in R, (a + b) \cdot c = (a \cdot c) + (b \cdot c)$$

As for groups, a ring R is said to be a commutative ring *if for any two elements $a, b \in R$, $a \cdot b = b \cdot a$.*

As an example, $\mathbb{Z}/n\mathbb{Z}$ is a commutative ring.

Definition 4. *Let G a set such that $G \neq \emptyset$ and let $+$ and $\times$ two binary relations on G. $(G, +, \times)$ is said to be a* field *if :*

- $(G, +)$ *is a commutative group of neutral element* 0 *;*
- $(G \setminus \{0\}, \times)$ *is a commutative group ;*
- $\times$ *is distributive over $+$: $\forall x, y, z \in G\ x \times (y + z) = (x \times y) + (x \times z)$ and the commutative version*

As an example, $\mathbb{Z}/p\mathbb{Z}$, where p is a prime number, is a field.

Definition 5. *The* order of a group G *is the number of its elements if the set if finite (or ∞ otherwise). It is written $\#G$.*

The order of an element $x \in G$ *is the order of the subgroup of G generated by x. If the order s of x is finite, then s is the smallest non-negative integer such that $s \cdot x = \underbrace{x + \cdots + x}_{s \text{ times}} = 0$.*

The characteristic of a ring $(R, +, \cdot)$ *with identity 1_R is the order of 1_R in its additive group $(R, +)$, that is to say the smallest non-negative integer s such that $s \cdot 1_R = \underbrace{1_R + \cdots + 1_R}_{s \text{ times}} = 0_R$.*

A field which has a finite number of elements is naturally called a *finite field*, and has an order q that is either prime or some power of a prime number. A finite field of order q is noted[1] $\mathbb{F}_q$. Unlike groups and rings, any finite field is known to be commutative (Wedderburn's little theorem).

Finite field theory is a rich domain that has at least a journal of its own[2].

Definition 6. *Given a finite field $\mathbb{F}_p$ with p a prime and an integer $m > 1$, the field $\mathbb{F}_{p^m}$ is called an* extension field *of the subfield $\mathbb{F}_p$.*

Elements $A \in \mathbb{F}_{q^m}$ can be represented by polynomials $A = a_{m-1}x^{m-1} + \cdots + a_0 x^0$ with $a_i \in \mathbb{F}_p$ for $i = 0, \ldots, m-1$.

[1] Sage [27] uses the notation `GF(q)`, which stands for "Galois field".

[2] http://www.journals.elsevier.com/finite-fields-and-their-applications/

2.3 Quadratic field

Definition 7. *Algebraic integers of the form* $a + b\sqrt{d}$*, where* d *is not a square, form a* quadratic field *which is written* $\mathbb{Q}[\sqrt{d}]$*. If* $d > 0$*, the field is called a* real quadratic field *and if* $d < 0$*, it is called an* imaginary quadratic field.

Definition 8. *The* discriminant *of the quadratic field* $\mathbb{Q}[\sqrt{d}]$ *is*

$$D = \begin{cases} d & \text{if } d \equiv 1 \pmod 4 \\ 4d & \text{otherwise} \end{cases}$$

Chapter 3

Elliptic Curves

Elliptic curves on finite fields (see fig. 3.2) are of particular interest, far beyond their origins: they were instrumental in proving Fermat's last theorem, they lie at the core of the Birch-Swinnerton-Dyer conjecture (one of the Millenium Prize problems), power algorithms for fast factorization of integers, provide methods of sphere packing in high dimensions and, last but not least, elliptic curves are key to modern cryptography.

In this chapter, we define elliptic curves and some related notions. We do not aim at providing a complete overview of this topic, which is incredibly large, but rather focus on some key aspects underpinning our discussion.

3.1 General definitions

There are many different definitions for elliptic curves, one of which is :

Definition 9. *An elliptic curve over a field $\mathbb{K}$ is a non-singular complete curve of genus 1, with a distinguished point "at infinity".*

When the characteristic of $\mathbb{K}$ is not 2 or 3, as is the case in the following discussion, it can be realized as a plane projective curve :
An *elliptic curve* E over $\mathbb{K}$ is defined by an equation of the form $y^2 = x^3 + ax + b$, $a, b \in \mathbb{K}$ (Weierstrass equation), with a distinguished point P_∞[1].

Definition 10. *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$, and r a number prime to q such than $r \mid \#E(\mathbb{F}_q)$, and $k \in \mathbb{N}$ minimal with $r \mid (q^k - 1)$. The number k is called the* embedding degree with respect to r *for the curve.*

In some contexts, k is also called the security multiplier *[24].*

Definition 11. *An* endomorphism ϕ *over an elliptic curve E in $\mathbb{F}_q$ is a map $E \to E$. An* efficient endomorphism *is an endomorphism that can be computed fast, as compared to point doubling. The* Frobenius map*:*

$$\pi : (x, y) \mapsto (x^q, y^q)$$

is an efficient endomorphism.

[1] Some authors use other notations for this point: 0, O, $\mathcal{O}$, ∞...

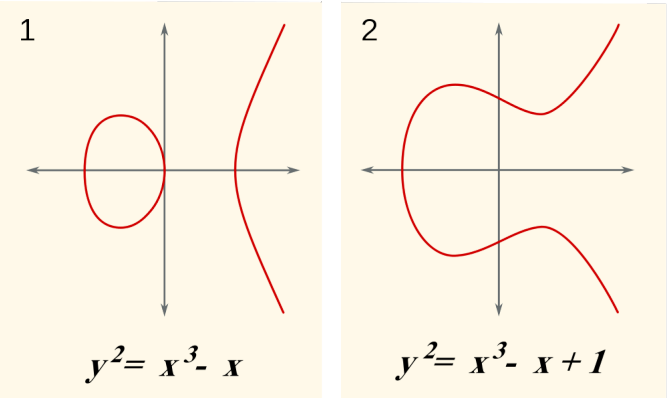

Figure 3.1: Two different elliptic curves considered in $\mathbb{R}$.

3.2 Abelian group structure

An operation of addition can be defined over elliptic curves, equipping them with an *abelian group structure*, where the point at infinity is the identity.

3.2.1 Addition operation

Consider two points P and Q of an elliptic curve E. Computing $P + Q$ on E can be understood geometrically.

In the general case, One can draw the line l through P and Q, then l crosses E at an other point R. $P + Q$ is obtained by taking the (vertical) line between R and P_∞.

There are a few special cases that have to be dealt with. These cases are represented in figure 3.2.1 :

- if $P = Q$, the line through P and Q is the tangent to the curve E in P. The rest of the computation is similar.
- if the line l through P and Q does not cross E in a third point, we introduce the "point at infinity" P_∞ and we consider that $P + Q = P_\infty$.
- if $Q = P_\infty$ (the symmetric case works the same), $P + P_\infty = P$.
- if l is tangent to E in Q, then $R = Q$.

We obtain a group $(E(\mathbb{F}), +)$ where the point P_∞ at infinity is the neutral element and where the inverse of a point P is its vertically symmetric point on the curve.

We can then compute $n \cdot P = \underbrace{P + \cdots + P}_{n \text{ times}}$ with P a point on the considered elliptic curve and n an integer. This operation, known as point mutiplication, is crucial in many applications and we devote chapter 4 to a method that computes it efficiently.

Definition 12. *Given $m \in \mathbb{Z}$, P is called an m-*torsion point *of E if it is a point of order m in E. That is to say : $m \cdot P = 0$ with 0 the neutral element.*

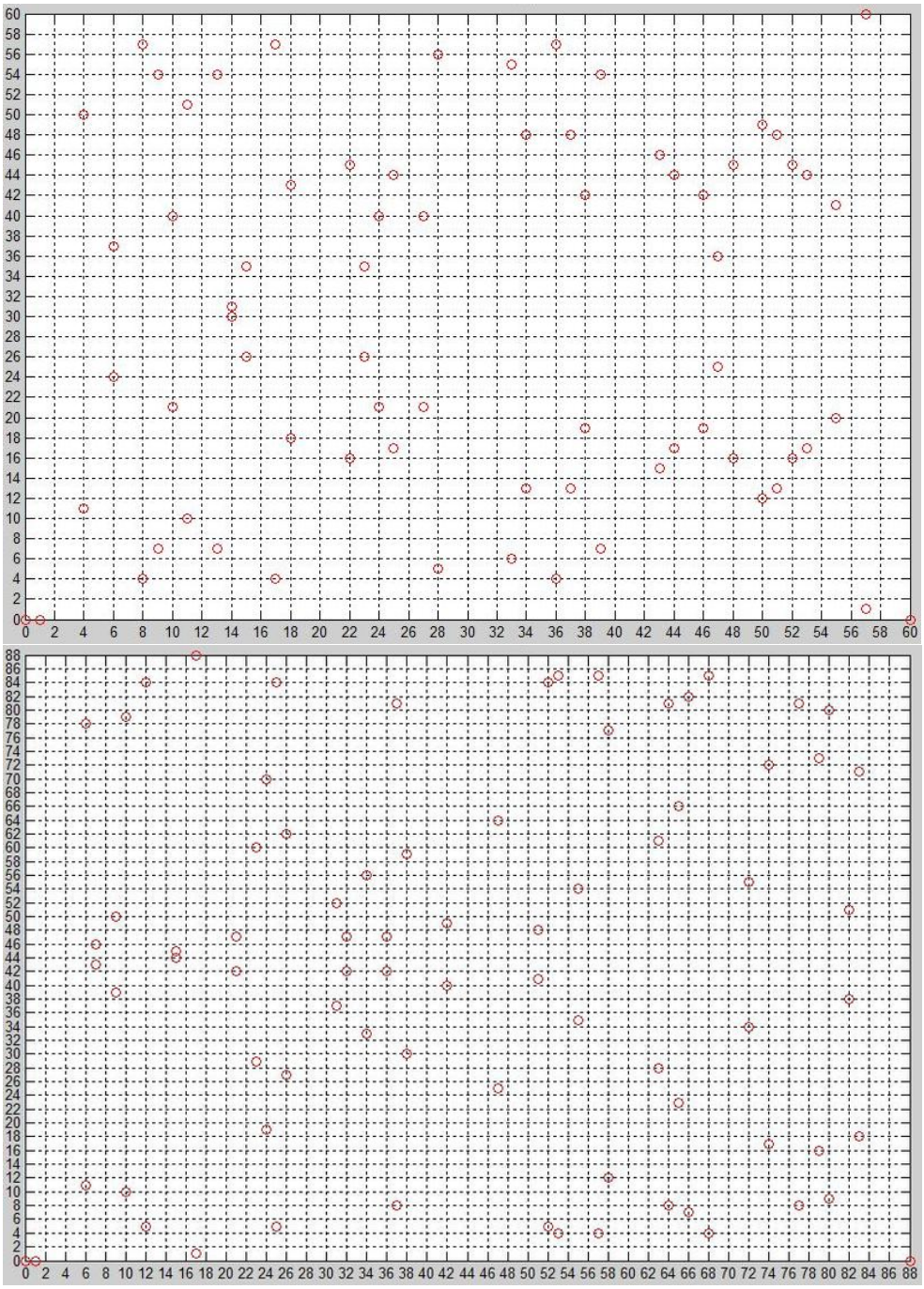

Figure 3.2: The same elliptic curve ($y^2 = x^3 - x$) over two different finite fields: $\mathbb{F}_{61}$ and $\mathbb{F}_{89}$.

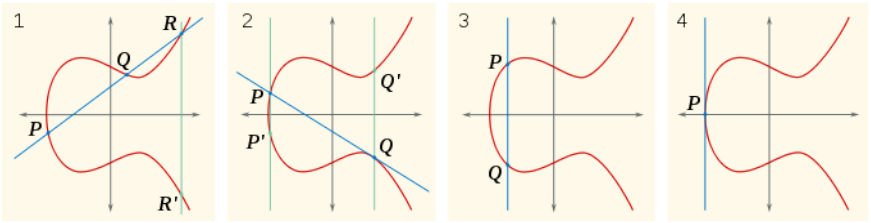

Figure 3.3: Sums on an elliptic curve. Four different cases are presented, the general one (1), when $Q = R$ (2), when $P = -Q$ (3) and when $P = Q = R$ (4)

The m-torsion subgroup of E, *denoted by* $E[m]$, *is the set of* m*-torsion points of* E.

When we consider elliptic curves on finite field, the construction of the addition carries on, with divisors (see section 3.3) and functions playing the role of lines.

3.3 Divisors

Definition 13. *A* divisor $D \in \mathrm{Div}(E)$ *is a formal sum of points on the elliptic curve* E *:*

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

with $a_P \in \mathbb{Z}$ *and* $a_P = 0$ *except for finitely many* $P \in E$. *The* degree *of* D *is the number*

$$\deg D = \sum_{P \in E} a_P$$

Naturally, the divisor of a product is the sum of the divisors, so that $\mathrm{Div}(E)$ has a group structure.

Definition 14. *Given* f *a function defined on the elliptic curve* E, *the* divisor of f *is :*

$$\mathrm{div}(f) = \sum_{P \in E} \mathrm{ord}_P(f)(P)$$

where $\mathrm{ord}_P(f)$ *is the order of the zero (or the pole) which* f *has at the point* P *and it is positive if* P *is a zero and negative otherwise.*

Definition 15. *Let* D *be a divisor. If there exists a function* f *such that* $D = \mathrm{div}(f)$, *then* D *is said to be a* principal divisor. *Principal divisors are of degree* 0.

The set of principal divisors is a subgroup of the group of divisors.

If $D = \sum_{p \in E} a_P(P)$ *is a principal divisor, for any function* f *such that* $\mathrm{div}(f)$ *and* D *have disjoint supports, we define* $f(D)$ *as*

$$f(D) = \prod_{P \in E} f(P)^{a_P}$$

As an example, if $ax + by + c = 0$ is the equation of a line passing through points A and B (where $A \neq \pm B$), crossing the curve at a third point C, then $f(x, y) = ax + by + c$ has exactly 3 zeroes, which are exactly A, B and C, and a pole of order three at infinity. Therefore the divisor of f is $D = (A) + (B) + (C) - E(P_\infty)$. Naturally, D is principal.

Definition 16. *Let A and B be two divisors. They are said to be* (linearly) equivalent*, and we write $A \sim B$, when their difference $A - B$ is a principal divisor, that is to say:*

$$A \sim B \quad \Leftrightarrow \quad \exists f, A = B + \mathrm{div}(f)$$

3.4 Pairings

At first, pairings were introduced as a tentative approach to solve the discrete logarithm problem (DLP) by Menezes, Okamoto and Vanstone [19]. The Tate or Weil pairings (see below) for instance map the discrete logarithm in a subgroup G of the group of points of an elliptic curve $E(\mathbb{F}_q)$ to the discrete logarithm in $\mathbb{F}_{q^k}$, where more efficient methods can be used: this is the basis of the Frey-Rück attack [9].

Pairings have since found many interesting applications in building cryptographic solutions, such as extensions of the Diffie-Hellman protocol for multiple parties (see section 5.1), identity-based cryptography (see section 5.2), self-blindable credentials [30] or short signatures [4].

We describe the Weil and Tate pairings. The first one is historically relevant, but has poor performance compared to the Tate pairing, on which our work focuses.

3.4.1 Definitions

Definition 17. *Given $\mathbb{G}_1$ and $\mathbb{G}_2$ groups noted additively and $\mathbb{G}_T$ a group noted multiplicatively, there exist P and Q generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. A* pairing *is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that has the following properties :*

- *it is bilinear : $\forall a, b,\ e(aP, bQ) = e(P, Q)^{ab}$*
- *it is non-degenerate : $e(P, Q) \neq 1$*

3.4.2 Weil pairing

Definition 18. *Let E be an elliptic curve defined over a field $\mathbb{K}$ and $l \in \mathbb{Z}$ which is prime to* $\mathrm{char}(\mathbb{K})$*. Let P, Q be two l-torsion points on the curve and D_P, D_Q two divisors with disjoint support such that:*

$$D_P \sim (P) - (P_\infty) \quad \textit{and} \quad D_Q \sim (Q) - (P_\infty)$$

There are two functions $f_{l,P}$ and $f_{l,Q}$ such that $\mathrm{div}(f_{l,P}) = lD_P$ *and* $\mathrm{div}(f_{l,Q}) = lD_Q$*. The* Weil pairing *is given by:*

$$e_l(P, Q) = \frac{f_{l,P}(D_Q)}{f_{l,Q}(D_P)}$$

Note that the functions $f_{l,P}$ and $f_{l,Q}$ are unique up to a constant and that the value of the pairing does not depend on the choice of these functions.

Example

Consider the elliptic curve $E : y^2 = x^3 + 3x$ defined on $\mathbb{F}_{11}$. The group structure of $E(\mathbb{F}_{11})$ is cyclic, therefore all points are linearly dependent, therefore the Weil pairing of any two points of this curve is 1.

Now let's consider the quadratic extension $\mathbb{F}_{11^2}$, which is obtained by defining the endomorphism

$$\begin{aligned} \phi : \mathbb{F}_{11^2} &\to \mathbb{F}_{11^2} \\ (x, y) &\mapsto (-x, iy) \end{aligned}$$

where $i^2 = -1$. If P is a point of $E(\mathbb{F}_{11})$ different from P_∞, then one can check that $\phi(P) \notin E(\mathbb{F}_{11})$. Hence P and $\phi(P)$ are linearly independent. If P is an m-torsion point, then

$$\begin{aligned} m \cdot P &= P_\infty \\ m \cdot \phi(P) &= \phi(m \cdot P) \\ &= \phi(P_\infty) \\ &= P_\infty \end{aligned}$$

which proves that $\phi(P)$ is also an m-torsion point (ϕ is a *distortion point*). We can therefore compute the Weil pairing of P and $\phi(P)$. For instance, let's take

$$\begin{aligned} P &= (3, 5) \\ Q &= \phi(P) \\ &= (-3, 5i) \\ &= (8, 5i) \end{aligned}$$

and observe that P (and Q) are of order 3. We further choose

$$\begin{aligned} P' &= (6, 6) \\ Q' &= (9, 6i) \end{aligned}$$

yielding different points P', $P + P'$, T' and $T + T'$. In order to compute the Weil pairing, we need two rational functions f_A and f_B such that

$$\begin{aligned} \mathrm{div}(f_A) &= 3(P + P') - 3(P') \\ \mathrm{div}(f_B) &= 3(Q + Q') - 3(Q') \end{aligned}$$

For instance,

$$\begin{aligned} f_A &= \frac{(y + 2x + 7)(y + 5x + 8)x}{(x + 10)(y + 10x)^2} \\ f_B &= \frac{(y + 2ix + 4i)(y + 5ix + 3i)}{(y + 4ix + 2i)(y + 8ix + 10i)} \end{aligned}$$

Now the Weil pairing can be computed in a straightforward way:

$$\begin{aligned} e_3(P, Q) &= \frac{f_A(Q + Q')f_B(P')}{f_A(Q')f_B(P + P')} \\ &= \frac{f_A(4, 10i)f_B(6, 6)}{f_A(9, 6i)f_B(7, 1)} \\ &= 5 + 8i \end{aligned}$$

Observe that $(5 + 8i)^3 = 1$: $e_3(P, Q)$ is a third root of unity in $\mathbb{F}_{11^2}$.

Properties

Theorem 1. *Let P, P_1, P_2 and Q, Q_1, Q_2 be n-torsion points of E, the Weil pairing e_n satisfies the following properties:*

- *bilinearity:*

$$e_n(P_1 + P_2, Q) = e_n(P_1, Q)e_n(P_2, Q)$$
$$e_n(P, Q_1 + Q_2) = e_n(P, Q_1)e_n(P, Q_2)$$

- *alternation:*

$$e_n(P, Q) = e_n(Q, P)^{-1}$$

- *non-degeneracy:*

$$\text{if } e_n(P, Q) = 1 \text{ for all } Q \text{ then } P = P_\infty$$

- *compatibility:*

$$P \in E[nm], Q \in E[n] \Rightarrow e_{nm}(P, Q) = e_n(m \cdot P, Q)$$

3.4.3 Tate pairing

In the following, we write $\mathbb{F}_q^* = \mathbb{F}_q \setminus \{0\}$. Let p be a prime number and $q = p^m$ for some positive integer m. We consider an elliptic curve E over a field $\mathbb{F}_q$.

Let $E[l]$ be the l-torsion points of E, with l coprime to q, $l \mid \#E(\mathbb{F}_q)$. Let k the embedding degree of E, that satisfies

$$l \mid (q^k - 1) \quad \text{and} \quad l \nmid (q^s - 1) \text{ for } 0 < s < k$$

Let $P \in E[l], Q \in E(\mathbb{F}_{q^k})/lE(\mathbb{F}_{q^k})$, let's define

$$\mathcal{A}_P \sim (P) - (P_\infty)$$
$$\mathcal{A}_Q \sim (Q) - (P_\infty)$$

and a function f_P such that $\text{div}(f_P) = n\mathcal{A}_Q$.

Definition 19. *The* Tate pairing *of P and Q is defined as follows:*

$$t_l(P, Q) = f_P(\mathcal{A}_Q)$$

Note however that this number is very often raised to the power $(q^k - 1)/l$ so as to obtain a unique value. This value is often called the modified Tate pairing.

As mentionned in the section devoted to divisors, this definition makes sense only if $\mathcal{A}_Q$ and $\text{div}(f_P)$ have disjoint supports. In order to ensure this property, let's choose $R \in E(\mathbb{F}_{q^k})[l]$. Then the following lemma holds:

Lemma 1.

$$t(P, Q) = \frac{f_P(Q + R)}{f_P(S)}$$

Proof. Since $1(Q+R)-1(R)$ has degree 0, $\sum a_P \cdot P = 1\cdot(Q+R)-1\cdot R = Q+R-R=Q$. Therefore:

$$\begin{aligned}(Q+R)-(R) &\sim (Q)-(P_\infty)\\ &\sim \mathcal{A}_Q\end{aligned}$$

We can thus define the Tate pairing as

$$t(P,Q)=\frac{f_P(Q+R)}{f_P(S)}$$

This formula holds Assuming that $\pm R$ and $\pm(Q+R)$ are not in the support of $\mathrm{div}(f_P)$, which happens with negligible probability. □

Example

As previously, consider the elliptic curve $E: y^2 = x^3+3x$ over $\mathbb{F}_{11}$. We use Miller's algorithm for the Tate pairing (see section 5.6), with points

$$\begin{aligned}P &= (1,9)\\ Q &= \phi(P)\\ &= (10,9i)\end{aligned}$$

and $l=6$. Indeed $6\,|\,(11^2-1)$ but $6\nmid(11-1)$).

We find a value of the Tate pairing equal to $2+7i$. In order to get a unique representant, this value has to be raised to the power $(11^2-1)/6=20$. In that case, $(2+7i)^{20}=5+3i$, which is a third root of unity.

Properties

Theorem 2. *Let $P,P_1,P_2\in E(\mathbb{F}_q)[l]$ and $Q,Q_1,Q_2\in E(\mathbb{F}_{q^k})[l]$. The Tate pairing t_l satisfies the following properties:*

- *bilinearity:*

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

- *non-degeneracy:*

$$\text{if } t_l(P,Q)=1 \text{ for all } Q \text{ then } P=P_\infty$$

- *compatibility:*

$$P\in E(\mathbb{F}_q)[l], Q\in E(\mathbb{F}_{q^k})[hl] \Rightarrow t_{hl}(h\cdot P,Q)=t_l(P,Q)^h$$

The following lemmas are immediate consequences of this:

Lemma 2. *For each $P\neq P_\infty$, there exists $Q\in E(\mathbb{F}_{q^k})[l]$ so that $t_l(P,Q)\neq 1$.*

Lemma 3. *Let $P\in E(\mathbb{F}_q)[l]$, $Q\in E(\mathbb{F}_{q^k})[l]$. For any $a\in\mathbb{Z}$,*

$$t_l(a\cdot P,Q)=t_l(P,Q)^a=t_l(P,\cdot Q)$$

Bilinear pairings from the Tate pairing

$E[l]$ is not a cyclic group of order l. This deficiency can be remedied in two ways [18]:

- If E is supersingular with $k > 1$, one can always choose a point P of order l and and endomorphism ψ such that $\phi(E) \notin \langle E \rangle$ (a distortion map), an define $\hat{e}(Q, R) = e(Q, \psi(R))$ defined over $\langle P \rangle \times \langle P \rangle$.
- If E is ordinary and $k > 1$, no such distortion map exists. However, one can choose points $P \in E(\mathbb{F}_q)$ and $Q \notin E(\mathbb{F}_q)$ of order l, and define $\hat{e}$ over $\langle P \rangle \times \langle Q \rangle$ by the restriction $\hat{e}(P, Q) = e(P, Q)$.

3.5 Complex multiplication

Complex multiplication of elliptic curves was developped in the early 20th century, and connects together several ideas in Algebra: imaginary quadratic fields, modular functions, endomorphism rings... This theory is especially rich in that it provides a bridge from abstract algebra to practical, implementable, algorithms. In fact, the only general and efficient algorithms to generate elliptic curves with prescribed properties (as the one we use in the following chapters) ultimately come from the theory of complex multiplication [7].

3.5.1 Isogenies and endomorphisms

Definition 20. *An* isogeny *between curves E_1 and E_2 is a morphism that sends P_∞ (on E_1) to P_∞ (on E_2). Such a mapping is a group homomorphism. The set of isogenies forms a group under addition which is denoted* $\mathrm{Hom}(E_1, E_2)$. *The set* $\mathrm{End}(E) = \mathrm{Hom}(E, E)$ *becomes a ring under the multiplication law given by composition. Elements of* $\mathrm{End}(E)$ *are called* endomorphisms of E.

For each $m \in \mathbb{Z}$, we can define the *multiplication by m* map

$$\begin{aligned} [m] : E &\to E \\ P &\to m \cdot P \end{aligned}$$

This map is an isogeny. Therefore there is an injection

$$[\,] : \mathbb{Z} \hookrightarrow \mathrm{End}(E)$$

When this map is not an isomorphism, that is to say $\mathbb{Z} \subset \mathrm{End}(E)$ strictly, we say that E *has complex multiplication.*

3.5.2 Example

Let $E : y^2 = x^3 + x$ be an elliptic curve over some field. Aside multiplication by integers, one can define

$$\begin{aligned} [i] : E &\to E \\ (x, y) &\mapsto (-x, iy) \end{aligned}$$

where $i^2 = -1$. This endomorphism is a genuine new element $[i]$ (that can also be noted $[\sqrt{-1}]$), so that E has complex multiplication by $\mathbb{Q}[i]$.

3.5.3 Finite fields, Frobenius endomorphism and the Hasse bound

The Frobenius endomorphism on elliptic curves defined over a finite field is also an endomorphism that differs from multiplication. Therefore all elliptic curves on finite fields have complex multiplication.

Hasse showed that the Frobenius endomorphism π of $\mathbb{F}_q$ satisfies the quadratic equation

$$\pi^2 - t\pi + q = 0$$

with discriminant $D = t^2 - 4q \leq 0$ and t is called the *trace of the Frobenius endomorphism*. Since the $\mathbb{F}_q$-rational points $E(\mathbb{F}_q)$ are precisely $\mathrm{Ker}(\pi - \mathrm{id})$, and $\pi - \mathrm{id}$ is separable, Hasse deduced that the cardinal of $E(\mathbb{F}_q)$ is given by $N(\pi - 1) = q + 1 - t$. This gives the *Hasse bound*:

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

Definition 21. *An elliptic curve E defined over a field of characteristic p is said to be* supersingular *when $p \mid t$. Otherwise it is said* non-supersingular*, or* NSS.

As an example, the curve $E : y^2 + y = x^3 + x + 1$ defined over $\mathbb{F}_2$ is supersingular [18].

3.5.4 Constructing curves

The main use of complex multiplication in cryptography is that it enabled the construction of elliptic curves with known properties, for instance the number of points can be known in advance.

Let p be a prime number and $q = p^m$, let $D < 0$. By Hasse's theorem,

$$4q = t^2 - u^2 D$$

has a solution with integers t and u. Therefore the cardinal of elliptic curves over $\mathbb{F}_q$ with complex multiplication by $\mathbb{Q}[\sqrt{D}]$ is

$$\#E(\mathbb{F}_q) = q + 1 - t$$

Building on this observation, several algorithms such as the one attributed to Cocks and Pinch (see section 5.3.2) have been constructed.

Chapter 4

Fast point multiplication using efficient endomorphisms

4.1 Motivational example: Diffie-Hellman key exchange

Point multiplication – the operation of computing $k \cdot P$ – is a core operation in most cryptographic protocols that use elliptic curves. A fast computation is needed for practical use, especially in key-agreement protocols such as Diffie-Hellman key exchange.

This protocol provides a way to construct a shared key, that can then be used for communication between two individuals, Alice and Bob. Alice has a key a, Bob has a secret key b. They keep this information to themselves, and agree on a group generator g, for instance $g \in E[\mathbf{F}_q]$ for some elliptic curve. Alice generates a public key $A = a \cdot g$, Bob generates a public key $B = b \cdot g$.

They share their respective shared keys with one another. Alice can now compute $a \cdot B = a \cdot (b \cdot g) = ab \cdot g$ and Bob can compute $b \cdot A = b \cdot (a \cdot g) = ba \cdot g = ab \cdot g$. This group element is the shared key.

The security of this protocol relies of the hardness of the discrete logarithm problem (DLP): given P and $x \cdot P$, it should be computationally hard to find x.

Alice	Public zone	Bob
$a = 18$		$b = 5$
	$p = 29, g = 10$	
	$A = 5, B = 8$	
$B^a = 22$		$A^b = 22$

Table 4.1: Example of the D-H key exchange protocol on $\mathbb{G} = \mathbb{Z}/p\mathbb{Z}^\times = \langle g \rangle^\times$. Both parties agree on $k = 22$.

In fact, the hardness of the DLP depends on the choice of the underlying group: we ought to use groups for which it is believed the DLP is indeed hard to solve. This is why elliptic curves are interesting in such applications, because there is no known generic algorithm that performs better than exponential-time in general.

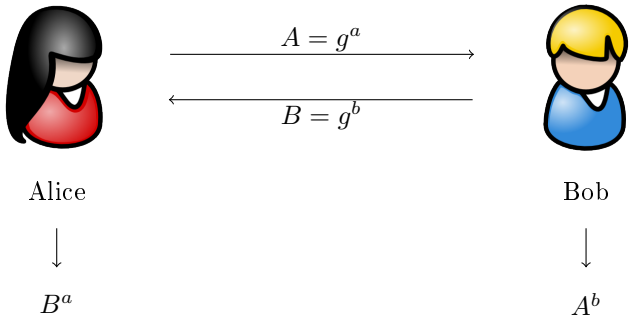

Figure 4.1: Diffie-Hellman key exchange agreement protocol, see table 4.1.

4.2 Double-and-add method

A simple method to compute $k \cdot P$ is to write k in its binary form and perform essentially point-doubling operations – computing $2P$ – and additions. This method simply builds on the rewriting

$$(n_i, \ldots, n_0)_2 \cdot P = n_i(2 \times 2^{i-1}) \cdot P + (n_i - 1, \ldots, n_0)_2 \cdot P$$

that leads to a straightforward, recursive algorithm. It can be implemented in interative form.

Algorithm 1 Double-and-add algorithm for point multiplication

$P_1 \leftarrow P$
$P_2 \leftarrow n_0 \cdot P$
for $j = i \downarrow 0$ **do**
    $n_i \leftarrow$ the i-th number of k in binary form
    $P_1 \leftarrow 2 \cdot P_1$
    if $n_i = 1$ **then**
        $P_2 \leftarrow P_2 + P_1$
    end if
end for
return P_2

If the binary form of k is sparse (low Hamming weight), then there are few point-doubling and additions to actually compute. At any rate, computing $k \cdot P$ for *small* values k is much more efficient than for large values of k.

4.3 The Gallant-Lambert-Vanstone method

The Gallant-Lambert-Vanstone method [11], or GLV method, is another technique aimed at simplifying point multiplication that builds on the observation

above. It relies on the existence of an *efficient endomorphism* of the curve, that is to say a rational map $\phi : E[\mathbb{F}_q] \to E[\mathbb{F}_q]$ that can be evaluated "fast", as compared to point-doubling and uses this endomorphism to break a big multiplication into smaller, faster ones.

Endomorphisms on elliptic curves include the negation map $P \mapsto -P$, the multiplication map $P \mapsto k \cdot P$ and the Frobenius map $P = (x, y) \mapsto (x^q, y^q)$. Exponentiating to the q-th power is a linear operation in $\mathbb{F}_{q^n}$, therefore the computation of the Frobenius map is indeed fast and many methods rely on this fact. The GLV method enables one to use general endomorphisms as well in order to speed up computation.

4.4 Presentation of the GLV algorithm

Suppose we have to compute a point multiplication $k \cdot P$, for a given integer k and point P on the elliptic curve E, and suppose we have an efficient endomorphism ϕ that satifies $\phi(P) = \lambda P$ for some λ. In the following, ϕ satisfies $\chi(\phi) = 0$ where $\chi(X) = X^2 + rX + s$ has λ as one of its roots.

4.4.1 First stage: decomposition of k

The first stage of the algorithm is to decompose the problem as follows:

$$\begin{aligned} k \cdot P &= (k_1 + k_2\lambda) \cdot P \\ &= k_1 \cdot P + k_2\lambda \cdot P \\ &= k_1 \cdot P + k_2 \cdot \phi(P) \end{aligned}$$

We need to find k_1 and k_2. In order to be efficient, these numbers both have to be as "small" as possible. We note $f : (i, j) \mapsto i + \lambda j$ and $\mathcal{K} = \ker f$, which is a sublattice of $\mathbb{Z} \times \mathbb{Z}$. The problem is to find b_1, b_2 two integers and v_1, v_2 two linearly independent vectors of $\mathcal{K}$ such that

$$(u_1, u_2) = (k, 0) - (b_1 v_1 + b_2 v_2)$$

is "short". Then if we set $(k_1, k_2) = u$ we have $k = k_1 + \lambda k_2$, and the decomposition is found.

Theoretical bounds on the decomposition

Sica *et al.* [25] observe that this leaves some room to define this criterion on u_1 and u_2 as minimizing

$$|(u_1, u_2)| = \max\{|u_1|, |u_2|\}$$

on which the original GLV paper focuses, leading to the "shortest" vector and found using an Euclidean algorithm, or minimizing

$$\langle (u_1, u_2) \rangle = \sqrt{u_1^2 + s u_2^2 - r u_1 u_2}$$

which gives the "smallest" vector. This last norm comes from the group homomorphism $(i, j) \mapsto i + j\phi \in \mathbb{Z}[\phi]$, which is a normed ring [25]. These two norms

are related but the last one has a straightforward geometrical interpretation, and they use an embedding of $\mathbb{Z}[\phi]$ in $\mathbb{C}$ to prove the following tight bound [25, Lemma 2, Theorem 3, Theorem 4] :

$$|u| < \mathbf{k}\sqrt{n}$$

where $\Delta < 0$ is the discriminant of P,

$$\mathbf{k} = \frac{2R\sqrt{s}}{\sqrt{|\Delta|}} > \frac{\sqrt{-\Delta}}{4}$$

and

$$R = \begin{cases} \frac{\sqrt{6-\Delta-\Delta^{-1}}}{4} & \text{if } r \text{ is odd,} \\ \frac{\sqrt{4-\Delta}}{4} & \text{if } r \text{ is even.,} \end{cases}$$

As a consequence, the GLV method can only produce short vectors when Δ is small and $\mathbf{k} \leq 1$.

Gaussian algorithm for lattice reduction

In the original paper [11], an Euclidean algorithm is used to find the values k_1 and k_2. Sica *et al.* [25] suggest the use of a Gaussian reduction algorithm, which has average constant time.

We give here a description of this algorithm. Given a lattice over the complex plane $\mathcal{L} = \{\lambda u + \mu v | \lambda, \mu \in \mathbb{Z}\}$, where (u, v) is a pair of $\mathbb{R}$-linearly independent elements of $\mathbb{C}$ called a basis. The Gaussian algorithm for lattice reduction provides a way to find a *reduced* basis (see fig. 4.2), which is made of two short vectors.

Figure 4.2: A lattice and two of its bases. The first basis is skew and the other one is reduced. [6]

We note $z = v/u$ and restrict our attention to lattices generated by a basis of the form $(1, z)$, a simple similarity transformation would yield the original lattice $\mathcal{L}$. Furthermore, define:

$$\mathcal{B} = \{z \in \mathbb{C} | |\Re(z)| \leq 1\}$$
$$\mathcal{D} = \left\{ z \in \mathbb{C} | \Re\left(\frac{1}{z}\right) \geq 1 \right\}$$

and the transformation

$$U : z \mapsto -\left(\frac{1}{z} - \left\lfloor \Re\left(\frac{1}{z}\right) \right\rfloor\right)$$

Given a complex number $z \in \mathcal{D}$, the following algorithm returns a value $z' \in \mathcal{B} \setminus \mathcal{D}$, so that $(1, z')$ is a minimal basis [6].

Algorithm 2 Centered Gaussian algorithm [6]

while $z \in \mathcal{D}$ **do**
  $z \leftarrow U(z)$
end while

4.4.2 Second stage: multi-exponentiation

The relevance of the previous decomposition is that simultaneous multiple point multiplication can be efficiently computed. Let u, v be integers represented in their binary form $(u_{t-1}, \ldots, u_0)_2$ and $(v_{t-1}, \ldots, v_0)_2$, P, Q be two points on an elliptic curve and $w \in \mathbb{N}$ the window width. The following algorithm computes $R = uP + vP$:

Algorithm 3 Simultaneous multiple point multiplication [11]

Compute $iP + jQ$ for all $i, j \in [0, 2^w - 1]$
$d \leftarrow \lceil t/w \rceil$
Write $u = (u^{d-1}, \ldots, u^0)$ where each u^i is a bitstring of length w
Write $v = (v^{d-1}, \ldots, v^0)$ where each v^i is a bitstring of length w
$R \leftarrow \infty$
for $i = d - 1 \searrow 0$ **do**
  $R \leftarrow Z^w R$
  $R \leftarrow R + (u^i P + v^i Q)$
end for
return R

Therefore, assuming that the bitlengths of k_1 and k_2 in the GLV method are indeed small, less point doubling operations occur than would have if kP was directly computed.

4.5 Examples

In the original GLV paper [11], several examples are provided so as to illustrate what is meant by *efficient* endomorphisms. It is noteworthy that these examples are exceptional in that $\mathbb{Z}[\phi]$ is principal and maximal [25] for all of them. Therefore there exists an element $\nu \in \mathbb{Z}[\phi]$ such that its norm is n, and $(\nu, \nu\phi')$ form a Gaussian reduced basis of $\mathcal{K}$.

Example 1 [25, Example 1][11, Example 3] Let $p \equiv 1 \pmod{n}$ be a prime. Define an elliptic curve E_1 over $\mathbb{F}_p$ by $y^2 = x^3 + ax$. The map $\phi : (x, y) \mapsto (-x, \beta y)$, where β is an element of order 4, is an endomorphism of E_1 that satisfies $\phi^2 + 1 = 0$. Therefore the theoretical bound on $u = (k_1, k_2)$ is $|u| \leq \sqrt{n/2}$.

Example 2 [25, Example 2][11, Example 4] Let $p \equiv 1 \pmod{3}$ be a prime and define an elliptic curve E_2 over $\mathbb{F}_p$ by $y^2 = x^3 + b$. Let γ be an element of order 3 and define the map $\phi : (x, y) \mapsto (\gamma x, y)$, which is an endomorphism of E_2 satisfying $\phi^2 + \phi + 1 = 0$. Therefore the theoretical bound on u is $|u| \leq \sqrt{7n}/3$.

Chapter 5

Pairing computation on elliptic curves with efficient endomorphisms

Scott [24] has designed a method for fast computing of pairings on some classes of elliptic curves of embedding degree 2. Ionică and Joux [13] extend this approach to classes of elliptic curves of small embedding degree (2, 4 and 6) which have an efficient endomorphism.

Indeed, the basic algorithm used in pairing computation was given by Miller (see section 5.6), and it is an extension of the double-and-add method for finding point multiples. Therefore, using efficient techniques for point multiplication (see chapter 4) leads to more efficient algorithms.

5.1 Motivational example 1: Diffie-Hellman revisited

In chapter 4 we described the Diffie-Hellman key exchange protocol. Suppose that three participants (A, B and C) want to agree on a common secret $K_{A,B,C}$ in a single pass: each participant is allowed to talk only once, and broadcast some data to the other two.

In standard D-H, all the participants choose a random number (their private key) that they keep to themselves: a, b and c. They agree on some elliptic curve E and some point P, and compute

$$P_A = a \cdot P$$
$$P_B = b \cdot P$$
$$P_C = c \cdot P$$

and broadcast these values (their public key). They can then use some function F such that

$$F(a, P_B, P_C) = F(b, P_A, P_C) = F(c, P_A, P_B) = K_{A,B,C}$$

The question is: how to find such a function F?

Using Tate pairings, Joux proposes the following method [14] (improved by Verheul [29]): choose two independent points P and Q and broadcast (P_A, Q_A), (P_B, Q_B) and (P_C, Q_C). For any two divisors D_1 and D_2, define

$$F(x, D_1, D_2) = t_n(D_1, D_2)^x$$

then A, B and C can respectively compute

$$\begin{aligned} K_1 &= F(a, (P_B) - (Q_B), (P_C + Q_C) - (P_\infty)) \\ K_2 &= F(b, (P_A) - (Q_A), (P_C + Q_C) - (P_\infty)) \\ K_3 &= F(c, (P_B) - (Q_B), (P_A + Q_A) - (P_\infty)) \end{aligned}$$

Thanks to the properties of the Tate pairing, the order in which the two other parties broadcast their public keys is not important

$$\begin{aligned} K_1 &= F(a, (P_C) - (Q_C), (P_B + Q_B) - (P_\infty)) \\ K_2 &= F(b, (P_C) - (Q_C), (P_A + Q_A) - (P_\infty)) \\ K_3 &= F(c, (P_A) - (Q_A), (P_B + Q_B) - (P_\infty)) \end{aligned}$$

These three keys are equal, their common value being

$$K_{A,B,C} = F(1, (P) - (Q), (P + Q) - (P_\infty))^{abc}$$

This method is not only an extension of D-H to three parties: thanks to pairings, it works in a single round, and opens the door to many refinements.

5.2 Motivational example 2 : Identity-based encryption

Public keys can be quite cumbersome to store and use: since they mostly look like a random sequence of numbers, there is no simple way to ensure that Bob's public key is indeed Bob's. A common solution is to invoke some third party that certifies a given key. This in turn requires a trustworthy certificate management, and some precautions need to be set.

Identity-based encryption (IBE) was proposed by Shamir as an open question in 1984: instead of meaningless sequences of numbers, can we use arbitrary character strings as public keys?

For instance, Alice would use "`bob@company.com`" to encrypt a message destined to Bob. There is no need for Alice to obtain any certificate. When Bob receives the message, he contacts a third party called the Private Key Generator (PKG). He authenticates himself to the PKG and the PKG gives Bob a private key he can use to decrypt the mail.

Of course, since the PKG knows Bob's private key, he has to be trusted to some degree. The first fully-functional, pratical identity-based encryption scheme was proposed by Boneh and Frank and relies on the Weil pairing [3].

Advantages and uses of such a system are very diverse and constitute an active research area:

- *Revocation of public keys*: certificates are generally valid until some expiration date – in an IBE system such a functionality can be realized by having Alice send Bob a message she encrypted with "`bob@company.org, current-year`". Each year, Bob would have to obtain a new private key from the PKG – Alice has nothing to do [3].
- *User credentials*: in the same fashion, managing user credentials is straightforward by having Alice use as a public key "`bob@company.com, current-year, clearance: secret`". The PKG grants or revokes user credentials, and manages key expiration.
- *Delegation*: suppose Alice encrypts a mail to Bob using the subject line as the key (e.g. "marketing", "human-resources"...), and suppose Bob has many assistants responsible for those different tasks. Then each assistant can decrypt the messages that falls within its responsibilities, but cannot decrypt messages intended for other assistants [3].

Essentially, IBE algorithms build on four algorithmic primitives:

- `setup`
  - Input: security parameter
  - Output: master key, public key
- `extract`
  - Input: master key, user ID
  - Output: user private key
- `encrypt`
  - Input: public key, user ID, message
  - Output: encrypted message
- `decrypt`
  - Input: user private key, encrypted message
  - Ouput: message (or error because of invalid key)

In Boneh and Franklin's original paper [3], they use Weil pairings to write the `encrypt` and `decrypt` primitives.

5.3 Generating pairing-friendly curves with embedding degree k

5.3.1 ρ-value of a curve

The best known algorithms for computing discrete logarithms on generic elliptic curves have complexity $O(\sqrt{r})$, where r is the order of the group. In the multiplicative group of a finite field, the index calculus algorithm solves the DLP in sub-exponential time. Therefore, in order to achieve the same level of security

in both the elliptic curve subgroups and in the finite field subgroup, we need to choose $q^k \gg r$, or equivalently we want $\rho \cdot k$ to be as small as possible, where the *ρ-value of a curve* is defined as [1, 13]

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

Small ρ-values may speed up arithmetic on the elliptic curve, while large ρ-values enable faster pairing evaluation. By the Hasse bound, $|\#E(\mathbb{F}_q) - q + 1| \leq 2\sqrt{q}$, ρ is always at least 1 (see [8, Table 8.2]).

5.3.2 The Cocks-Pinch method

In order for the DLP to be hard, elliptic curves should have large prime order subgroups. If chosen at random, such curves may have arbitrary embedding degree. For practical implementations, we need this degree to be small.

The Cocks-Pinch method, used in [13], constructs curves with a large prime order subgroup and a small value of the embedding degree. However, these curves have a $\rho - value$ close to 2: p is too large compared to r.

In order to generate a pairing friendly curve, we are looking for p, r, k, d, t and y such that

$$\begin{aligned} r \,|\, ny^2 + (t-2)^2 \\ r \,|\, p^k - 1 \\ t^2 + ny^2 = 4p \end{aligned}$$

If we choose a small k, r a prime number, a discriminant n and $k \,|\, (r-1)$, then the Cocks-Pinch algorithm provides the numbers p and t such that there is a curve over $\mathbb{F}_p$ with $p + 1 - t$ points, where $r \,|\, (p + 1 - t)$ and $r \,|\, (p^k - 1)$.

Algorithm 4 Cocks-Pinch algorithm

Choose g a primitive k-th root of unity in $\mathbb{F}_r$
Choose an integer $t \equiv g + 1 \pmod{r}$
if $\gcd(t, n) \neq 1$ **then**
    Choose another g
end if
Choose an integer $y_0 = \pm\frac{(t-2)}{\sqrt{-n}} \pmod{r}$
$j \leftarrow 0$
repeat
    $p \leftarrow (t^2 + n(y_0 + jr)^2)/4$
    $j \leftarrow j + 1$
until p is prime
return p and t

This method has several advantages [8] : it works for any k, for almost any CM discriminant and r is chosen in advance.

5.4 Preliminary results

We are working on an elliptic curve E defined over a finite field $\mathbb{F}_q$ with neutral element P_∞, we give ourselves a function $f_{k,Q}$ defined for any r-torsion point Q

and integer j with divisor

$$\mathrm{div}(f_{k,Q}) = k(Q) - (kQ) - (k-1)(P_\infty)$$

Lemma 4. *Let i and j two integers, let P an r-torsion point on E, let l be the line through iP and jP, intersecting E at one other point R, let v be the line through R and P_∞. Then we have the following results :*

$$\begin{aligned}
\mathrm{div}(f_{r,P}) &= r(P) - r(P_\infty) \\
\mathrm{div}(l) &= (iP) + (jP) + (R) - 3(P_\infty) \\
\mathrm{div}(v) &= (R) + ((i+j)P) - 2(P_\infty) \\
f_{i+j,P} &= f_{i,P} f_{j,P} \frac{l}{v}
\end{aligned}$$

Proof. The results are simple rewritings of divisors:

- *First result*

$$\begin{aligned}
\mathrm{div}(f_{r,P}) &= r(P) - (rP) - (r-1)(P_\infty) \\
&= r(P) - (P_\infty) - (r-1)(P_\infty) \text{ as P is an } r\text{-torsion point on } E
\end{aligned}$$

- *Second result*
  By definition, $\mathrm{div}(l) = \sum_{P \in E} \mathrm{ord}_P(l)(P)$. l is the line through iP, jP and it crosses E only once in R. All these zeros have order 1. Moreover, P_∞ is a pole of order three.

- *Third result*
  Similarly, v is the line through R and P_∞, therefore v crosses E in another point which is by definition $(i+j)P$. This gives $\mathrm{div}(v) = (P_\infty) + (R) + ((i+j)P) - 3(P_\infty)$

- *Fourth result*

$$\begin{aligned}
&\mathrm{div}(f_{i,P}) + \mathrm{div}(f_{j,P}) + \mathrm{div}(l) - \mathrm{div}(v) \\
&\quad = (i(P) - (iP) - (i-1)(P_\infty)) + (j(P) - (jP) - (j-1)(P_\infty)) + \\
&\quad ((iP) + (jP) + (R) - 3(P_\infty)) - ((R) + ((i+j)P) - 2(P_\infty)) \\
&\quad = (i+j)(P) - ((i+j)P) - (i+j-1)(P_\infty) \text{ after reorganization on terms} \\
&\quad = \mathrm{div}(f_{i+j,P})
\end{aligned}$$

□

Lemma 5. *Let m and n be two integers and P a point on E,*

$$f_{mn,P} = f_{m,P}^n f_{n,mP}$$

Proof.

$$\begin{aligned}
\mathrm{div}(f_{m,P}^n) + \mathrm{div}(f_{n,mP}) &= n\,(m(P) - (mP) - (m-1)(P_\infty)) + (n(mP) - (nmP) - (n-1)(P_\infty)) \\
&= nm(P) - (nmP) - (n(m-1) + (n-1))(P_\infty) \\
&= nm(P) - (nmP) - (nm-1)(P_\infty) \\
&= \mathrm{div}(f_{nm,P})
\end{aligned}$$

□

Theorem 3. *Let P be a point on E of order a prime integer $r \neq q$. Assuming $k \geq 1$, let Q a point over $\mathbb{F}_{q^k}$ such that $\pi(Q) = qQ$, Then P and Q are eigenvectors of any other endomorphism of E.*

5.5 Ionică-Joux pairing

In this section, we detail how to compute efficiently a Tate-like pairing on elliptic curves that have an efficient endomorphism ϕ, following the method introduced by Ionică and Joux [13].

Definition 22. *Let R_1 and R_2 be two points of an elliptic curve, we note their* correction *:*

$$corr_{R_1,R_2} = \frac{l_{R_1,R_2}}{v_{R_1+R_2}}$$

where l_{R_1,R_2} is the line through R_1 and R_2 and $v_{R_1+R_2}$ the vertical line through $R_1 + R_2$.

Lemma 6. *Let ϕ be an endomorphism of E of degree b, P and Q two points on the curve E. Then, for any integer λ the following equality is true up to a constant :*

$$f_{\lambda,\phi(P)}(\phi(Q)) = f^b_{\lambda,P}(Q) \left(\prod_{K \in \operatorname{Ker}\phi \setminus \{P_\infty\}} corr_{P,K}(Q) \right)^{-\lambda} \left(\prod_{K \in \operatorname{Ker}\phi \setminus \{P_\infty\}} corr_{\lambda P,K}(Q) \right)$$

Proof. The basic layout of the proof is given in [13], we provide additional details. We are interested in the zeros and poles of $\phi^*(f_{\lambda,\phi(P)})$. By definition,

$$\operatorname{div}(f_{\lambda,\phi(P)}) = \lambda(\phi(P)) - (\lambda\phi(P)) - (\lambda - 1)(P_\infty)$$

Therefore,

- Q is a zero of $\phi^*(f_{\lambda,\phi(P)})$ if and only if $\phi(Q)$ is a zero of $f_{\lambda,\phi(P)}$. There is only one such zero : $\phi(P)$ of order λ. Therefore, Q is a zero of $\phi^*(f_{\lambda,\phi(P)})$ if and only if $Q = P + K$ with $K \in \operatorname{Ker}\phi$. Similarly to $\phi(P)$, it has order λ.

- Q is a pole of $\phi^*(f_{\lambda,\phi(P)})$ if and only if $\phi(Q)$ is a pole of $f_{\lambda,\phi(P)}$. There are two possible such poles : $\lambda\phi(P)$ of order 1 and P_∞ of order $(\lambda - 1)$. Therefore, Q is a pole of $\phi^*(f_{\lambda,\phi(P)})$ if and only if $Q = \lambda P + K$ (of order 1) or $Q = K$ (of order $(\lambda - 1)$) with $K \in \operatorname{Ker}\phi$.

We know that

$$\operatorname{div}(l_{K,P}) - \operatorname{div}(v_{K+P}) = (K) + (P) - (K + P) - (P_\infty)$$

So that we have the following :

$$\begin{aligned}
\mathrm{div}(\phi^*(f_{\lambda,\phi(P)})) &= \sum_{K\in\mathrm{Ker}\,\phi} \lambda(P+K) - \sum_{K\in\mathrm{Ker}\,\phi} (\lambda P+K) - \sum_{K\in\mathrm{Ker}\,\phi} (\lambda-1)(K) \\
&= \lambda \sum_{K\in\mathrm{Ker}\,\phi} ((P+K)-(K)) - \sum_{K\in\mathrm{Ker}\,\phi} ((\lambda P+K)-(K)) \\
&= \lambda \sum_{K\in\mathrm{Ker}\,\phi} \left((P)-(P_\infty)-\mathrm{div}\left(\frac{l_{K,P}}{v_{K+P}}\right)\right) - \sum_{K\in\mathrm{Ker}\,\phi} \left((\lambda P)-(P_\infty)-\mathrm{div}\left(\frac{l_{K,\lambda P}}{v_{K+\lambda P}}\right)\right) \\
&= \lambda \sum_{K\in\mathrm{Ker}\,\phi} ((P)-(P_\infty)-\mathrm{div}(corr_{P,K})) - \sum_{K\in\mathrm{Ker}\,\phi} ((\lambda P)-(P_\infty)-\mathrm{div}(corr_{\lambda P,K})) \\
&= b\,(\lambda(P)-(\lambda P)-(\lambda-1)(P_\infty)) \sum_{K\in\mathrm{Ker}\,\phi} ((-\lambda\mathrm{div}(corr_{P,K})+\mathrm{div}(corr_{\lambda P,K})) \\
&= b\cdot\mathrm{div}(f_{\lambda,P}) - \lambda\cdot\mathrm{div}\left(\prod_{K\in\mathrm{Ker}\,\phi\setminus\{P_\infty\}} corr_{P,K}\right) + \mathrm{div}\left(\prod_{K\in\mathrm{Ker}\,\phi\setminus\{P_\infty\}} corr_{\lambda P,K}\right)
\end{aligned}$$

□

Theorem 4. *Let E be an elliptic curve over a finite field $\mathbb{F}_q$, r a prime number such that $r\,|\,\#E(\mathbb{F}_q)$, k the embedding degree with respect to r. Let ϕ be an efficiently computable separable endomorphism of E, whose characteristic equation is $X^2+aX+b=0$.*

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be the subgroups of order r whose elements are eigenvectors of ϕ defined over $\mathbb{F}(q)$ and $\mathbb{F}(q^k)$ respectively. Let λ be the eigenvalue of ϕ on $\mathbb{G}_1$, verifying $\lambda^2+a\lambda+b=cr$ with $r\nmid bc$. Then the map $a_\phi(.,.):\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r$ is a bilinear non-degenerate pairing :

$$\begin{aligned}
a_\phi(P,Q) = f_{\lambda,P}^{\lambda+a}(bQ)\ f_{\lambda,P}^{b}(\hat{\phi}(Q))\ f_{a,\lambda P}(bQ)\ f_{b,P}(bQ)\times \\
\left(\prod_{K\in Ker\phi\setminus\{P_\infty\}} corr_{P,K}(\hat{\phi}(Q))\right)^{-\lambda} \left(\prod_{K\in Ker\phi\setminus\{P_\infty\}} corr_{\lambda P,K}(\hat{\phi}(Q))\right)\times \\
corr_{\lambda^2 P,a\lambda P}(bQ)\, l_{\lambda^2 P+a\lambda P,bP}(bQ)
\end{aligned}$$

A proof can be found in [13]. This last theorem provides us with a pairing linked to the Tate pairing.

5.6 Miller's algorithm

Let E be an elliptic curve given by $y^2=x^3+ax+b$, defined over a finite field $\mathbb{F}_q$. Let r be a large prime number dividing $\#E(\mathbb{F}_q)$ and k be the corresponding embedding degree. Let P be an r-torsion point, and denote by $f_{i,P}$ the function with divisor

$$\mathrm{div}(f_{i,P}) = i(P)-(iP)-(i-1)(P_\infty)$$

Note that $\mathrm{div}(f_{r,P}) = r(P)-r(\mathcal{O})$ (Lemma 1). Suppose we want to compute the sum of iP and jP: the line l connects these two point and crosses the curve

on a third point R, while the vertical line v connects R and the desired point. The corresponding divisors are

$$\begin{aligned}\mathrm{div}(l) &= (iP) + (jP) + (R) - 3(P_\infty) \\ \mathrm{div}(v) &= (R) + ((i+j)P) - 2(P_\infty)\end{aligned}$$

The following relation ("Miller's equation") holds

$$f_{i+j,P} = f_{i,P} f_{j,P} \frac{l}{v}$$

Suppose we want to compute $f_{r,P}(Q)$, this provides us with an iterative method: at step i, we can compute mP (where m is the integer with binary expansion given by the i topmost bits or r), and in the same time compute $f_{m,P}(Q)$ with the formula above. This is essentially Miller's algorithm for the Tate pairing $t_r(P, Q)$.

Algorithm 5 Miller's algorithm for the Tate pairing

$i \leftarrow \lfloor \log_2 r \rfloor$
$K \leftarrow P$
$f \leftarrow 1$
while $i \geq 1$ **do**
  Compute equations of l and v appearing in the doubling of K
  $K \leftarrow 2K$
  $f \leftarrow f^2 \frac{l(Q)}{v(Q)}$
  if the i-th bit of r is 1 **then**
    Compute equations of l and v that appear in the addition $P + K$
    $K \leftarrow P + K$
    $f \leftarrow f \frac{l(Q)}{v(Q)}$
  end if
  $i \leftarrow i - 1$
end while
return f

Chapter 6

Efficient pairing computation for Barreto-Naehrig curves of embedding degree $k = 12$

In this chapter, we consider a certain family of elliptic curves introduced by Barreto and Naehrig in 2006 [1] which have an embedding degree $k = 12$. These curves are especially well suited for pairing-based cryptography. We extend the method of Scott [24] and Ionică-Joux [13] to efficient pairing computation on these curves.

6.1 Barreto-Naehrig curves

Barreto and Naehrig introduced a new type of curves [1] :

Definition 23. *A* BN curve *is an elliptic curve E_b of the form $y^2 = x^3 + b$ defined over $\mathbb{F}_p$ such that $b \neq 0$ and $n = \#E_b(\mathbb{F}_p)$ and n, q ad t (the trace of the frobenius) are given by :*

$$\begin{aligned} q &= q(u) = 36u^4 + 36u^3 + 24u^2 + 6u + 1 \\ n &= n(u) = 36u^4 + 36u^3 + 18u^2 + 6u + 1 \\ t &= t(u) = 6x^2 + 1 \end{aligned}$$

Such a curve has embedding degree 12.

Whereas the Cocks-Pinch method (see section 5.3.2) gives curves with $\rho \sim 2$, the BN construction gives curves with $\rho \sim 1$ which is a lot more efficient.

Example (160 bits BN curve of prime order p and $k = 12$) [1]

$$\begin{aligned} p &= 1461501624496790265145448589920785493717258890819 \\ n &= 1461501624496790265145447380994971188499300027613 \\ t &= 120892581430521795886320 7 \\ b &= 3 \\ y &= 2 \end{aligned}$$

Algorithm 6 Barreto-Naehrig algorithm [1] (see appendix A)

m is the approximate size desired for the curve order (in bits)
$P(x) \equiv 36x^4 + 36x^3 + 24x^2 + 6x + 1$
Compute the smallest $x \approx 2^{m/4}$ such that $\lceil \log_2 P(-x) \rceil = m$
loop
    $t \leftarrow 6x^2 + 1$
    $p \leftarrow P(-x)$
    $n \leftarrow p + 1 - t$
    if p and n are prime **then**
        exit loop
    end if
    $p \leftarrow P(x)$
    $n \leftarrow p + 1 - t$
    if p and n are prime **then**
        exit loop
    end if
    $x \leftarrow x + 1$
end loop
$b \leftarrow 0$
repeat
    repeat
        $b \leftarrow b + 1$
    until $b + 1$ is a quadratic residue mod p
    Compute y such that $y^2 = b + 1 \pmod{p}$
    $G \leftarrow (1, y)$ on the curve $E : y^2 = x^3 + b$
until $nG = P_\infty$
return p, n, b, y

6.2 Efficient endomorphims on BN curves

We have an obvious endomorphism on a BN curve E of equation $y^2 = x^3 + b$:

$$\phi : (x, y) \to (\beta x, y)$$

with β a non-trivial cube root of unity. Its characteristic equation is

$$\phi^2 + \phi + 1 = 0$$

An eigenvalue λ verifies the equation $\lambda^2 + \lambda + 1 = cr$.

Example As an example, we consider the curve of the previous section E : $y^2 = x^3 + 3$ defined over the finite field $\mathbb{F}_p$ where

$$p = 146150162449679026514544858992078549371725889081 9$$

A non-trivial root of unity in this field is

$$\beta = 2923003248995208495450923854321783187178438239430$$

Corresponding values of λ, c and r are:

$$r = n = 146150162449679026514544738099497118849930002761 3$$
$$c = 2$$
$$\lambda = 1876592702571924201401286942037491663606564352 60$$

computed with Sage.

6.3 Pairing computation

Pairings can now be computed, keeping in mind that the conditions $r \,|\, \#E(\mathbb{F}_q)$ and $r \nmid bc$ are satisfied, using the main theorem of section 5.5 along with the natural efficien endomorphism of BN curves given in the section 6.2.

The result of this computation is a power of the Tate pairing, and the computation itselfs takes advantage of the endomorphism.

Example On the curve of the previous section, which has embedding degree $k = 12$, we consider two points of order n :

$P = (1, 2)$

$Q = (32559303200497221829814784474794291 77, 146150162449679026514544738099497118849930002761 1)$

Performing the full computation on these points yields their Ionică-Joux pairing :

$$a(P, Q) = 2923003248996836460610944484172923855446079484403$$

6.4 Future work and perspectives

We provided a construction that enables efficient pairing computation on Barreto-Naehrig elliptic curves with embedding degree 12, which results in speeding up a key operation in cryptographic protocols, on curves that have interesting mathematical properties from a security perspective. This work focused on a variant of the Tate pairing developped by Ionică and Joux.

Other pairings (Ate and Eta pairings) exist, and might be relevant. However we consider some other directions to be especially interesting, and recent works could certainly enrich our approach.

In [22], the notion of skew Frobenius map is introduced. We have already mentionned in 3.5.3 the Frobenius endomorphism. The computation can be improved by working in the dual world when there exist known endomorphism that allow to go from normal operations on curves to their duals. We have only concentrated on elliptic curves and pairings that could be defined directly on these curves, maybe that we could gain by also working with duals.

In [23], the key idea is to parallelize the computation of a particular pairing by writing it as products of two different functions that do not depend on each other. This idea has been applied successfully on BN curves and on some curves of degree 8 which has lead to algorithms allowing to compute pairings. In both cases a speed-up in the computation of pairings has been observed. An interesting way of dwelling further into this problem would be to apply the same idea to parallelize the algorithm we have developped based on [13] (if possible).

These two ideas rely on further constructions. We have introduced the GLV method in 4. It is used by Scott and Galbraith in [10] to work more quickly directly in $\mathbb{G}_2$ and $\mathbb{G}_T$ when computing a pairing $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. This should be easily applied in order to improve the time needed for computing pairings.

Bibliography

[1] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected areas in cryptography*, pages 319–331. Springer, 2006.

[2] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. pages 354–368. Springer-Verlag, 2002.

[3] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology—CRYPTO 2001*, pages 213–229. Springer, 2001.

[4] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Advances in Cryptology—ASIACRYPT 2001*, pages 514–532, 2001.

[5] H. Cohen, G. Frey, and R. Avanzi. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC press, 2006.

[6] H. Daudé, P. Flajolet, and B. Vallée. An average-case analysis of the gaussian algorithm for lattice reduction. *Combinatorics, Probability and Computing*, 6(04):397–433, 1997.

[7] A. Enge. Courbes algébriques et cryptologie. Habilitation 'a diriger des recherches, Université Paris-Diderot - Paris VII, 2007.

[8] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010.

[9] G. Frey and H.G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of computation*, 62(206):865–874, 1994.

[10] S. Galbraith and M. Scott. Exponentiation in pairing-friendly groups using homomorphisms. *Pairing-Based Cryptography–Pairing 2008*, pages 211–224, 2008.

[11] R. Gallant, R. Lambert, and S. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In *Advances in Cryptology – CRYPTO 2001*, pages 190–200. Springer, 2001.

[12] S. Goldwasser and M. Bellare. Lecture notes on cryptography. *Summer course "Cryptography and computer security" at MIT*, 1999:1999, 1996.

[13] S. Ionică and A. Joux. Pairing computation on elliptic curves with efficiently computable endomorphism and small embedding degree. *Pairing-Based Cryptography-Pairing 2010*, pages 435–449, 2010.

[14] A. Joux. A one round protocol for tripartite diffie-hellman. *Algorithmic number theory*, pages 385–393, 2000.

[15] A. Joux. The weil and tate pairings as building blocks for public key cryptosystems. *Algorithmic number theory*, pages 11–18, 2002.

[16] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[17] M. Maas. *Pairing-based cryptography*. PhD thesis, Master's Thesis, Technische Universiteit Eindhoven, 2004.

[18] A. Menezes. An introduction to pairing-based cryptography. *Department of Combinatorics and Optimazion*, 2005.

[19] A.J. Menezes, T. Okamoto, and S.A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *Information Theory, IEEE Transactions on*, 39(5):1639–1646, 1993.

[20] V. Miller. Use of elliptic curves in cryptography. In Hugh Williams, editor, *Advances in Cryptology – CRYPTO '85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin / Heidelberg, 1986.

[21] G.C.C.F. Pereira, M.A. Simplício, M. Naehrig, and P.S.L.M. Barreto. A family of implementation-friendly bn elliptic curves. *Journal of Systems and Software*, 2011.

[22] Y. Sakemi, Y. Nogami, K. Okeya, H. Kato, and Y. Morikawa. Skew frobenius map and efficient scalar multiplication for pairing–based cryptography. *Cryptology and Network Security*, pages 226–239, 2008.

[23] Y. Sakemi, S. Takeuchi, Y. Nogami, and Y. Morikawa. Accelerating twisted ate pairing with frobenius map, small scalar multiplication, and multi-pairing. *Information, Security and Cryptology–ICISC 2009*, pages 47–64, 2010.

[24] M. Scott. Faster pairings using an elliptic curve with an efficient endomorphism. *Progress in Cryptology – INDOCRYPT 2005*, pages 258–269, 2005.

[25] F. Sica, M. Ciet, and J.J. Quisquater. Analysis of the Gallant-Lambert-Vanstone method based on efficient endomorphisms: Elliptic and hyperelliptic curves. In *Selected areas in cryptography*, pages 21–36. Springer, 2003.

[26] J.H. Silverman. *The arithmetic of elliptic curves*, volume 106. Springer Verlag, 2009.

[27] W. A. Stein et al. *Sage Mathematics Software (Version 4.8.0)*. The Sage Development Team, 2012. http://www.sagemath.org.

[28] M. Stögbauer. Efficient algorithms for pairing-based cryptosystems. *Germany: Darmstadt University of Technology*, 2004.

[29] E. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. *Advances in Cryptology – EUROCRYPT 2001*, pages 195–210, 2001.

[30] E. R. Verheul. Self-blindable credential certificates from the weil pairing. pages 533–551. Springer-Verlag, 2001.

Appendix A

Implementation of the Barreto-Naehrig algorithm

We implemented the Barreto-Naehrig algorithm (see section 6.1) for elliptic curve generation in Sage [27]. It searches for an elliptic curve of order roughly m bits long, and outputs p, n, b, y, such that $y^2 = x^3 + b$ has order n over $\mathbb{F}_p$ and $G = (1, y)$ is a generator of the curve.

```
1  def getEllipticCurve(K, b):
2      '''
3      Construct an elliptic curve of the form
4        y^2 = x^3 + b
5      on the field K
6      '''
7      return EllipticCurve(K, [0, b]);
8
9  def BarretoNaehrig(m, max_iter = 1000):
10     '''
11     Constructs a [BN06] curve of order p, where
12     p is roughly m bits long
13     '''
14     def P(x):
15         return 36*(x)^4 + 36*(x)^3 + 24*(x)^2 + 6*(x) + 1;
16
17     print "Looking for x..."
18     var('x')
19     X = floor(find_root(ceil(log(P(-x))/log(2)) == m, 0, 2*2^(m/4)))
20
21     # ---------------------------------------------------------------
22
23     print "Looking for p and n..."
24     for i in range(max_iter):
25         t = 6 * X^2 + 1
26         p = P(-X)
27         n = p + 1 - t
28         if is_prime(p) and is_prime(n):
29             break
30         p = P(X)
31         n = p + 1 - t
32         if is_prime(p) and is_prime(n):
```

```
33              break
34          X = X + 1
35
36      if not (is_prime(p)) or not (is_prime(n)):
37          print "An error occured, p or n is not a prime"
38          print "Increase max_iter"
39          return "An error occurred"
40
41      # ------------------------------------------------------------
42
43      b = 0
44      while 1:
45          print "Starting loop"
46
47          # ------------------------------------------------------------
48
49          print "\tLooking for b..."
50          while b < p:
51              b = b + 1
52              number_of_roots = 1 + legendre_symbol(b+1, p)
53              if number_of_roots > 0:
54                  break
55          if b == p:
56              print "A problem occurred... No b found"
57              break
58          print "\tFound b = ", b
59
60          # ------------------------------------------------------------
61
62          print "\tLooking for y..."
63          found = False
64
65          for y in [1..p]:
66              mod1 = y^2 % p
67              mod2 = (b+1) % p
68              if (mod1 == mod2):
69                  found = True
70                  break
71
72          if not found:
73              print "A problem occurred... No y found"
74              break
75          print "\tFound y = ", y
76
77          # ------------------------------------------------------------
78
79          E = getEllipticCurve(GF(p), b)
80          G = E([1, y])
81          if n*G == 0*G:
82              print "Curve found !"
83              return p, n, b, y
84          print "nG != infinity, look for b larger than", b
85
86      return "An error occured"
87
88
89  # Generate curve parameters
90  p, n, b, y = BarretoNaehrig(10)
91  E = getEllipticCurve(GF(p), b)
```

```
92  P = E([1, y])
93  print E
94
95  # Checking
96  print E.cardinality()
97  print E.trace_of_frobenius()
98  print P.order()
99
100 # Mugshot
101 plot(E)
```