EFFICIENT ARITHMETIC ON HYPERELLIPTIC CURVES WITH REAL REPRESENTATION

David J. Mireles Morales

Royal Holloway and Bedford New college,

University of London

Thesis submitted to

The University of London

for the degree of

Doctor of Philosophy

2008.

Declaration

These doctoral studies were conducted under the supervision of Professor Steven D. Galbraith.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Mathematics Department as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

David José Mireles Morales

Abstract

We discuss arithmetic in the Jacobian of a hyperelliptic curve C of genus g. The traditional approach is to fix a point P at infinity in C and represent divisor classes in the form $E - dP$. We propose a different representation which is balanced at infinity. The resulting arithmetic is more efficient than previous approaches when there are 2 points at infinity.

The geometric framework presented in this analysis is then used to give a new interpretation of the infrastructure associated to a hyperelliptic curve. We prove that there is a natural injection from the set of infrastructure ideals to the class group of the corresponding curve, and use this result to give very precise results relating the difficulty of computing the distance of an arbitrary infrastructure ideal to the discrete logarithm problem in the curve.

The efficient arithmetic on hyperelliptic curves afforded by our proposal is used to efficiently implement pairings. We present several optimisation ideas that allow us to conclude that pairings can be efficiently computable in real models of hyperelliptic curves.

We present an attack on one of the hidden pairing schemes proposed by Dent and Galbraith. We drastically reduce the number of variables necessary to perform a multivariate attack and in some cases we can completely recover the private key. Our attack relies only on knowledge of the public system parameters.

Finally, we present ideas relating the pairing inversion problem and the discrete logarithm problem on an elliptic curve. This is done using the reduction from the DLP to the Diffie-Hellman problem developed by Boneh, Lipton, Maurer and Wolf. This approach fails when only one of the pairing inversion problems can be solved. In this case we use Cheon's algorithm to get a reduction.

Acknowledgements

I would like to thank Professor Steven Galbraith for his supervision and guidance. Steven has always been very generous, and he has helped me and encouraged me whenever I had a project or idea. Special thanks to my advisor Kenny Paterson for his support.

It is a pleasure to thank Kevin Buzzard, Victor Flynn, Mike Harrison, Kristin Lauter, Ben Smith and Fre Vercauteren for their help and advice. I would also like to thank the members of the Mathematics Department and the Information Security Group for the excellent research environment provided. Thanks to my friends Adrian, Jacob, and the inhabitants of the House of Crypto.

I want to thank my family for their continual support and encouragement. Thank you Rebe, because this wouldn't make sense without you.

Finally, I would like to thank Royal Holloway and Conacyt for their financial support.

Contents

Chapter 1

Introduction

1.1 Pairings

A very interesting consequence of the use of elliptic curves as building blocks for discrete log based cryptosystems has been the adoption by cryptographers of concepts once exclusive to the realms of number theory and algebraic geometry.

Among these concepts adopted by the cryptography community, the use of bilinear pairings has had impressive implications. Pairings first appeared in cryptography with the work of Menezes, Okamoto and Vanstone in [70]. The authors of [70] use the Weil pairing to reduce the discrete logarithm problem in the group of points of an elliptic curve E over $\mathbf{F}_q$, to the discrete logarithm problem in the group $\mathbf{F}_{q^k}^*$ for some k. Since there exists a sub-exponential algorithm to compute discrete logarithms in $\mathbf{F}_{q^k}^*$, if the value of k is not too large, this construction simplifies the computation of discrete logarithms in $E(\mathbf{F}_q)$.

The authors of [70] also prove that if E is a supersingular elliptic curve,

then the value of k (known as the embedding degree, see Definition 2.3.2) is bounded by $k \leq 6$, which automatically implied that the mov reduction could be applied on all supersingular elliptic curves. This sent supersingular elliptic curves (and in general all *pairing friendly* curves) to the backwaters of cryptography, since they were considered *weak*.

The rehabilitation of pairing friendly curves into mainstream cryptography started with the work of Joux [57], where a one round tripartite Diffie-Hellman algorithm is proposed. After the work of Joux, the number of constructive applications of pairings in cryptography has exploded. Perhaps the most famous application is the construction of an ID-based cryptosystem by Sakai, Ohgishi, and Kasahara [83] and Boneh and Franklin [7]. The concept of ID-based cryptography was introduced by Shamir in [89], but no efficient ID-based cryptosystem had been found until the publication of [83, 7].

The newly found applications of pairings to cryptography have kept cryptographers with a flair for number theory (and number theorists with a flair for cryptography) rather busy. For example, finding suitable curves and parameters for a given application is a lot harder. A good account of the pitfalls in the selection of curves is given in [34], and [28] presents parameters to achieve certain security levels.

Two very active areas of research have been the search for pairing-friendly curves construction algorithms, and the development of optimised pairing computation techniques, usually focused in computing pairings using short Miller loops.

According to [28], the constructions of pairing-friendly curves can be broadly divided into curves in families and curves not in families. Examples of constructions of curves in families are given by the work of Miyaji, Nakabayashi

and Takano [75], and its generalisations by Scott and Barreto [88] and Galbraith, McKee and Valença [40]. Examples of algorithms to find curves not in families are given by the Cocks-Pinch method (see Chapter IX of [5]) and the construction by Dupont, Enge and Morain [23].

The development of loop-shortening techniques has been equally successful. The original idea was presented by Duursma and Lee in [24], and was later extended by Barreto et.al. in [3], and by Hess et.al. in [49]. The existence of these loop-shortening techniques has prompted some questions regarding the difficulty of the discrete logarithm problem in curves where pairings can be computed with very short loops as described in [37].

1.2 Higher genus curves

Another trend in mathematical cryptography has been an attempt to extend results and algorithms available for elliptic curves to higher genus hyperelliptic curves. One of the reasons to use hyperelliptic curves is that the number of $\mathbf{F}_q$-rational points on the Jacobian J of a hyperelliptic curve C of genus g defined over $\mathbf{F}_q$ grows as q^g, so in principle a smaller based field is necessary to achieve a similar security level as one would achieve with an elliptic curve. For pairing-based cryptosystems, hyperelliptic curves offer a wider range of parameters.

An important difference between elliptic and hyperelliptic curves is that there exist algorithms to solve the discrete logarithm problem in the Jacobian of a hyperelliptic curve that are faster than generic group algorithms [1, 43], whereas no such algorithm is known for elliptic curves. For these reason, hyperelliptic curves with high genus are considered weak for cryptography.

Cantor [13] and Koblitz [61] developed algorithms to compute in the Jacobian of a hyperelliptic curve given by a real model, and Paulus and Rück extended these algorithms to cover all hyperelliptic curves. Optimised explicit addition formulae have been found for genus 2 imaginary curves [17]. We will describe our work with hyperelliptic curves given by a real model in Chapter 3.

Some progress has also been done trying to find good point-counting algorithms for higher genus curves. Pila [81] extends Schoof's algorithm [87] to high dimension abelian varieties. Unfortunately, Pila's algorithm requires knowledge of certain explicit polynomials associated to the abelian variety, and computing these polynomials is still an open problem. Further progress in point-counting algorithms includes the work by Huang and Ierardi [52], Adleman and Huang [2] and Gaudry and Harley [44]. Additional techniques are available if the base field has small characteristic, as shown by Kedlaya [60].

There has also been considerable progress extending the theory of pairings to hyperelliptic curves. Frey and Rück [32] extend the MOV attach to hyperelliptic curves, and Galbraith [35] proved that the embedding degree of all supersingular curves of a given genus is bounded. The articles that laid the foundations for the loop-shortening techniques [24, 3, 49] all dealt with hyperelliptic curves, so no further work was necessary to adapt these techniques to higher genus. Pairings have been efficiently implemented in hyperelliptic curves given by an imaginary model [3, 77], we will discuss the implementation of pairings on hyperelliptic curves given by a real model in the following section.

The construction of pairing-friendly hyperelliptic curves with a given number of points has focused on the construction of genus 2 curves. This is a consequence of the existence of a theory for genus 2 curves analogous to the

theory of complex multiplication for elliptic curves [92]. This theory was developed by Igusa [53, 54], who described three polynomials, known as the Igusa polynomials, analogous to the Hilbert class polynomial, and associated three arithmetic invariants to every genus 2 hyperelliptic curve that are analogous to the j-invariant of an elliptic curve. Just as in the case of elliptic curves, one needs to compute the Igusa polynomials in order to construct hyperelliptic curves with a given number of points using an extension of the CM-method. There are several approaches to the problem of computing the Igusa class polynomials [25, 45, 97], and once the Igusa invariants of a curve are known, an algorithm of Mestre [71] shows that one can recover an equation of the curve. An algorithm extending the Cocks-Pinch method to genus 2 curves was given by Freeman [27], and by Freeman, Stevenhagen and Streng [29].

1.3 Dissertation outline

The motivation for the results in this dissertation comes from the intersection of the two trends described in the previous sections. The original observation was the fact that no-one had implemented pairings on real hyperelliptic curves, despite the fact that with very high probability all the hyperelliptic curves produced with the CM-method over a field k would only have a k-rational real model and no k-rational imaginary model.

While studying the possibilities for the implementation of pairings on hyperelliptic curves given by a real model, it became clear that the representation of elements in the Jacobian of the curve used in the literature [26, 55, 79, 80] was not optimal. This led us to the representation of elements in the Jacobian (or equivalently, in the class group) presented in Chapter 3, and published

in [36].

This representation allows for faster arithmetic in the class group of a hyperelliptic curve given by a real model, with the advantage that all the explicit formulae developed to deal with real hyperelliptic curves [26] can still be used. Finally, we give a divisor inversion algorithm, a fundamental tool in any efficiently computable group that had surprisingly been overlooked in the literature.

Once the theoretical framework for arithmetic in the Jacobian of a hyperelliptic curve given by a real model had been developed, we set out to implement pairings efficiently in a case amenable to comparison with pairings in an elliptic curve. The hyperelliptic curve was provided by [41], and the implementation was made in collaboration with S. Galbraith and X. Lin. Our pairing implementation uses a recently developed loop-shortening technique, the R-ate pairing, developed by Lee, Lee and Park in [64]. Our implementation includes some novel optimisation techniques and a new algorithm to find appropriate curve parameters. All the details are presented in Chapter 5 and will be published as [39].

The geometric insight gained while working in the material of Chapter 3 then allowed us to give a geometric interpretation of the set of infrastructure ideals in a real function field. This interpretation is presented in Chapter 4 and the article outlining these ideas has been submitted for publication [72]. One of the most important consequences of the interpretation of infrastructure that we develop is that we can prove the equivalence between the problem of finding the distance of a given infrastructure ideal and the discrete logarithm problem in the class group of the corresponding curve. We have also been

able to explain precisely why the infrastructure has 'holes', and we can describe exactly where these 'holes' will appear. As a further consequence of our work, we can prove in a very precise way that using hyperelliptic curves with the representation we describe in Chapter 3 is more efficient than the use of infrastructure for group-based cryptography.

Chapters 6 and 7 also deal with questions related to pairing based cryptography, although the results they present are independent from the rest of this thesis. In Chapter 6, we describe an attack on a cryptosystem proposed by Dent and Galbraith in [19]. Our proposal applies to schemes with minimal functionality, and in the general case dramatically reduces the number of variables necessary to perform an attack on the scheme. In some special cases our attack recovers the secret key of the scheme using only linear algebra. This second attack can be seen as a warning against careless implementations. The results of Chapter 6 have been published in [74].

Chapter 7 analyses the consequences of the existence of pairing-inversion algorithms. This is motivated by the results of [37], who prove that for elliptic curves with a very reduced Miller loop, the 'Miller inversion' problem can be solved efficiently. In Chapter 7, we analyse the theoretical and practical implications that the existence of an efficient (or relatively efficient) pairing inversion algorithm would have. The results of this chapter are available as [73].

Chapter 2

Mathematical Background

In this chapter we present the mathematical concepts that will be used in the later chapters of this thesis. We begin with a section describing the basic properties of algebraic curves, in Section 2.1 we focus on results regarding the divisor class group and the Riemann-Roch theorem. In Section 2.2 we present the basic properties of hyperelliptic curves. We conclude with Section 2.3, where the Tate- and Weil-pairings are defined, we prove some of their basic properties and present results allowing for the computation of pairings using short Miller loops.

2.1 Curves

Throughout this thesis, a curve will be a nonsingular projective curve defined over a perfect field k. A point on a curve C is a geometric point on C defined over $\overline{k}$.

2.1.1 The Group of Divisors

Definition 2.1.1. *Let C be an algebraic curve defined over a field k. The* group of divisors *on C is the free abelian group generated by the points of $C(\overline{k})$. It is denoted as* $\mathrm{Div}(C)$.

In other words, $\mathrm{Div}(C)$ is the group of finite formal sums $D = \sum n_i P_i$, for integers n_i and points P_i on $C(\overline{k})$.

Definition 2.1.2. *A divisor $D = \sum n_i P_i$ is said to be* effective *if every coefficient n_i is non-negative. Given two divisors D_1 and D_2, we will say that $D_1 \geq D_2$ if $D_1 - D_2$ is an effective divisor.*

Definition 2.1.3. *The support of the divisor $D = \sum n_i P_i$ is the set of points P_i for which the coefficient n_i is nonzero. Note that the support of a divisor is a finite set.*

Given a divisor $D = \sum_i n_i P_i$, we say that the divisor $D_z = \sum_i \max(n_i, 0) P_i$, is the *divisor of zeros* of D. Analogously, the divisor $D_p = \sum_i \min(n_i, 0) P_i$, is the *divisor of poles* of D. Definition 2.1.7 in the next section explains the origin of this nomenclature.

Definition 2.1.4. *Given a divisor D, we define its degree* $\deg D$ *as*

$$\begin{aligned} \deg : \mathrm{Div}(C) &\longrightarrow \mathbb{Z} \\ \sum n_i P_i &\mapsto \sum n_i. \end{aligned}$$

The degree function is a homomorphism from the group of divisors into $\mathbb{Z}$.

The set of degree 0 divisors, denoted $\mathrm{Div}^0(C)$, is a subgroup of $\mathrm{Div}(C)$, as it is the kernel of this morphism.

2.1.2 Principal Divisors

Given a curve C defined over the field k, let $\overline{k}(C)$ define the field of $\overline{k}$-valued rational functions on C.

Definition 2.1.5. *Let P be a point on the curve C. Let $\mathcal{O}_P$ denote the ring of elements of $\overline{k}(C)$ which are well defined at P. This is the* local ring of P on C. *It is a discrete valuation ring (see [33]), with maximal ideal $m_P = \{f \in \mathcal{O}_P | f(P) = 0\}$.*

We denote $\mathrm{ord}_P : \mathcal{O}_P - \{0\} \longrightarrow \mathbb{Z}$ for the normalized valuation on $\mathcal{O}_P$, assigning to every function f in $\mathcal{O}$ the (non-negative) integer n such that $f \in m_P^n\{\}m_P^{n+1}$ (such an integer exists because $\mathcal{O}$ is a discrete valuation ring). We naturally extend this valuation to the function field $\overline{k}(C)^*$. The properties of this valuation are given by the following lemma.

Lemma 2.1.6. *The function* ord_P *has the following properties:*

1. $\mathrm{ord}_P(fg) = \mathrm{ord}_P(f) + \mathrm{ord}_P(g)$ *for all* $f, g \in \overline{k}(C)$.
2. *For a fixed function $f \in \overline{k}(C)$, there are only finitely many points P on C with* $\mathrm{ord}_P(f) \neq 0$.
3. *Let $f \in \overline{k}(C)$. Then* $\mathrm{ord}_P(f) \geq 0$ *if and only if $f \in \mathcal{O}_P$. Similarly,* $\mathrm{ord}_P(f) > 0$ *if and only if $f \in m_P$.*
4. *For $f \in \mathcal{O}_P$, the following are equivalent:*
   - $\mathrm{ord}_P(f) \geq 0$ *for all P.*
   - $\mathrm{ord}_P = 0$ *for all P.*

- $f \in k^*$.

Definition 2.1.7. *Let C be a curve, and let $f \in \overline{k}(C)^*$ be a nonzero rational function. The* divisor associated *to f is the divisor*

$$\operatorname{div}(f) = \sum_{P \in C(\overline{k})} \operatorname{ord}_P(f).$$

This is a well defined divisor since Lemma 2.1.6 shows that for a fixed function f there are only finitely many points P on C such that $\operatorname{ord}_P(f) \neq 0$.

Definition 2.1.8. *We say that a divisor D is* principal *if it is the divisor associated to a function $D = \operatorname{div}(f)$. We will denote the group of principal divisors on C as* $\operatorname{Prin}(C)$.

Lemma 2.1.9. *Every principal divisor has degree 0.*

Proof. See Hartshorne [47][Corollary II.6.10]. □

Definition 2.1.10. *Two divisors D_0 and D_1 are* linearly equivalent*, denoted $D_0 \equiv D_1$, if there is a function $f \in \overline{k}(C)^*$ such that*

$$\operatorname{div}(f) = D_1 - D_0.$$

Remark 2.1.11. The concept of linear equivalence on divisors is fundamental in the study of curves. We will briefly explain the origin of the term. Let D_1 and D_2 be two divisors such that $D_1 = D_2 + \operatorname{div}(f)$. For each point $(x : y)$ on the projective line $\mathbf{P}^1$, define a divisor $D_{(x:y)} = D_2 + \operatorname{div}(x + yf)$. Then $D_{(1:0)} = D_2$ and $D_{(0:1)} = D_1$. Our construction shows a family of divisors parametrized by the points of a line that deforms D_1 to D_2.

2.1.3 The Divisor Class Group

Definition 2.1.12. *The* divisor class group *of C is the group of divisor classes modulo linear equivalence. We will denote it as* $\mathrm{Cl}(C)$*. The class of a divisor D in* $\mathrm{Cl}(C)$ *will be denoted by* $[D]$*.*

By Lemma 2.1.9, every principal divisor has degree 0. It follows that the degree function is well defined on divisor classes. We define $\mathrm{Cl}^0(C)$ as the degree zero subgroup of $\mathrm{Cl}(C)$. Equivalently, $\mathrm{Cl}^0(C)$ can be defined as the set of degree zero divisors $\mathrm{Div}^0(C)$ modulo principal divisors.

Example 2.1.1. Let $E : y^2 = x^3 + Ax + B$ be an elliptic curve given in Weierstrass form. Given two points P and Q on E, let $l_{P,Q}$ denote the line passing through them. Denote the third intersection point of $l_{P,Q}$ with E as R, and let v_R be the vertical line passing through R. If $\mathcal{O}$ is the point at infinity of E, the divisor associated to the function $l_{P,Q}/v_R$ is

$$\mathrm{div}\left(\frac{l_{P,Q}}{v_R}\right) = P + Q - \overline{R} - \mathcal{O}, \tag{2.1.1}$$

where $\overline{R}$ is the elliptic conjugate of R. The standard chord-and-tangent addition algorithm on E says that $P + Q = \overline{R}$. If we define a function

$$\begin{aligned} \phi : E &\longrightarrow \mathrm{Cl}^0(E) \\ P &\mapsto [P - \mathcal{O}] \end{aligned}$$

Equation (2.1.1) can be rewritten as

$$[P - \mathcal{O}] + [Q - \mathcal{O}] = [\overline{R} - \mathcal{O}],$$

proving that ϕ is a group homomorphism. This shows that the chord-and-tangent group law on E coincides with the group operation in the class group of E.

Definition 2.1.13. *If the curve C is defined over a non algebraically closed field k, we say that a divisor D is k-rational if it is invariant under the action of the Galois group* $\mathrm{Gal}(\overline{k}/k)$. *The group of k-rational divisors is denoted as* $\mathrm{Div}_k(C)$.

Note that points in the support of a k-rational divisor D might not be k-rational. Analogously, we say that a divisor class is k-rational if it is $\mathrm{Gal}(\overline{k}/k)$-stable. We denote the group of k-rational divisor classes as $\mathrm{Cl}_k(C)$.

Lemma 2.1.14. *Let $D = \mathrm{div}(f)$ be the divisor of a function $f \in \overline{k}(C)$. If the divisor D is k-rational, there exists a function $f' \in k(C)$ such that $D = \mathrm{div}(f')$.*

Proof. Let K be a finite Galois extension of k such that f is K-rational. Let $\mathbf{G}_{K/k} = \mathrm{Gal}(K/k)$ denote the Galois group of K over k.

Since D is k-rational, given $\sigma \in \mathbf{G}_{K/k}$, the function $\sigma(f)/f$ has divisor $\mathrm{div}(\sigma(f)/f) = 0$. Lemma 2.1.6 implies that $\sigma(f)/f \in K^*$. A simple calculation shows that the function

$$\begin{aligned} \phi : \mathbf{G}_{K/k} &\longrightarrow K^* \\ \sigma &\mapsto \sigma(f)/f, \end{aligned}$$

is a 1-cocycle. Hilbert's Theorem 90 implies that it is a 1-coboundary. Hence, there exists $a \in K^*$ such that

$$\sigma(f)/f = \sigma(a)/a \quad \text{for every } \sigma \in \mathbf{G}_{K/k},$$

which can be rewritten as

$$\sigma(f/a) = f/a \quad \text{for every } \sigma \in \mathbf{G}_{K/k},$$

proving that f/a is a k-rational function with associated divisor D.

□

We defined linear equivalence for divisors on a curve C using $\overline{k}$-valued functions. Lemma 2.1.14 shows that linear equivalence of k-rational divisors can be defined using only k-rational functions. In other words, the natural map

$$i : \mathrm{Div}_k(C)/\operatorname{div}(k(C)^*) \longrightarrow \mathrm{Cl}(C),$$

is an injection.

Proposition 2.1.15. *Let C be a curve defined over the field k. A divisor class $[D] \in \mathrm{Cl}(C)$ is k-rational if and only if it contains a k-rational divisor.*

Proof. Let $\mathbf{G}_{\overline{k}/k} = \mathrm{Gal}(\overline{k}/k)$ denote the absolute Galois group of k. Clearly, if a divisor class contains a k-rational divisor it will be stable under the action of $\mathbf{G}_{\overline{k}/k}$.

To prove the converse let D be the representative of a $\mathbf{G}_{\overline{k}/k}$-stable divisor class. Since D has only finitely many points in its support, let K be a finite Galois extension of k over which all the points in the support of D are defined. Again, we denote $\mathbf{G}_{K/k} = \mathrm{Gal}(K/k)$ for the Galois group of K over k. Let P be a point on C that doesn't belong to the support of $\sigma(D)$ for every $\sigma \in \mathbf{G}_{K/k}$.

We define the map

$$\psi : \mathbf{G}_{K/k} \longrightarrow K(C)$$
$$\sigma \mapsto f,$$

which given σ, assigns it the unique function f in $K(C)$ with associated divisor $\mathrm{div}(f) = D - \sigma(D)$ and such that $f(P) = 1$. Such a function exists because the divisor class $[D]$ is $\mathbf{G}_{K/k}$-stable. The map ψ is a 1-coboundary, so Hilbert's Theorem 90 implies that it is a 1-cocycle, i.e. there exists a function f_ψ in $K(C)$ such that

$$\psi(\sigma) = \sigma(f_\psi)/f_\psi \text{ for every } \sigma \in \mathbf{G}_{K/k}.$$

This last relation can be rewritten as

$$\mathrm{div}(\sigma(f_\psi)/f_\psi) = D - \sigma(D),$$

for every $\sigma \in \mathbf{G}_{K/k}$. This is equivalent to

$$D + \mathrm{div}(f_\psi) = \sigma(D + \mathrm{div}(f_\psi)),$$

for every $\sigma \in \mathbf{G}_{K/k}$. In other words, the divisor $D + \mathrm{div}(f_\psi)$ is k-rational, and by definition it belongs to the class $[D]$. □

If the curve C is defined over a finite field $\mathbf{F}_q$ with q elements, the qth power Frobenius automorphism π_q acts on C. This action can be naturally extended to the group $\mathrm{Div}(C)$ and is well defined in the quotient $\mathrm{Cl}(C)$. Throughout this thesis we will abuse notation and denote this action as π_q.

2.1.4 The Riemann-Roch Theorem

Definition 2.1.16. *Let C be a curve defined over a field k, and let D be a divisor on C. The set $L_k(D)$ is defined as*

$$L_k(D) = \{f \in k(C)^* | D + \mathrm{div}(f) \geq 0\} \cup \{0\}.$$

The set $L_k(D)$ is a finite dimensional k-vector space, we will denote its dimension as $l(D)$.

If the divisor D is k-rational, the dimension of the vector space $L_k(D)$ is the same as the dimension of the vector space $L_{\overline{k}}(D)$ [50][Proposition A.2.2.10]. Hence, there is a natural identification $L_{\overline{k}}(D) = \overline{k} \otimes L_k(D)$. Since the dimension of the space $L_k(D)$ is independent of the field k (provided the divisor D is k-rational), we do not need to specify the field k in the following theorem.

Theorem 2.1.17 (Riemann-Roch theorem)**.** *Let C be a curve. There exists a divisor K_C on C and an integer $g \geq 0$ such that for all divisors $D \in \mathrm{Div}(C)$,*

$$l(D) - l(K_C - D) = \deg(D) - g + 1.$$

Proof. See Hartshorne [47][Theorem IV.1.3]. □

The divisor K_C is called the canonical divisor on C. This definition applies to every divisor linearly equivalent to K_C.

Definition 2.1.18. *The integer g is called the* genus *of the curve C.*

Corollary 2.1.19. *Let C be a curve of genus g. Then $l(K_C) = g$ and $\deg(K_C) = 2g - 2$.*

Proof. Using Riemann-Roch with the divisor $D = 0$ we get $1-l(K_C) = -g+1$, which proves the first formula. Now, if we use Riemann-Roch with $D = K_C$ we get $l(K_C) - 1 = \deg(K_C) - g + 1$. □

Example 2.1.2. We will use the Riemann-Roch Theorem to prove a classic result. Let C be a curve of genus g and fix a degree g divisor D_g on C. Take any divisor class $[D]$ in $\mathrm{Cl}^0(C)$ with representative D. The Riemann-Roch Theorem implies that $l(D_g - D) \geq 1$, so there exists a function f such that the divisor $D_0 := D_g - D + \mathrm{div}(f)$ is an effective divisor. This can also be expressed as $[D_0 - D_g] = [D]$. In other words, we have proved that every divisor class in $\mathrm{Cl}^0(C)$ has a representative of the form $D_0 - D_g$, where D_0 is an effective divisor.

Let f be a function on the curve C, and let $D = \sum_{i \in I} n_i P_i$ be a divisor. If the supports of D and $\mathrm{div}(f)$ are disjoint, we define

$$f(D) = \prod_{i \in I} f(P_i)^{n_i}.$$

Theorem 2.1.20 (Weil reciprocity)**.** *Let f and g be functions on C whose divisors have disjoint supports. Then*

$$f(\mathrm{div}(g)) = g(\mathrm{div}(f))$$

2.2 Hyperelliptic Curves

Definition 2.2.1. *A curve C of genus $g \geq 2$ is a hyperelliptic curve if it is a double covering of the projective line.*

Not every curve of genus $g \geq 2$ is hyperelliptic. One can prove however

that all genus 2 curves are hyperelliptic.

Theorem 2.2.2. *Every genus two curve C is hyperelliptic.*

Proof. Take an effective canonical divisor K_C. By the Riemann-Roch Theorem, the space $L(K_C)$ has dimension 2, hence it contains a non-constant function x with divisor of poles K_C. Hence, the map

$$\begin{aligned} \phi : C &\longrightarrow \mathbf{P}^1 \\ P &\mapsto (x(P) : 1) \end{aligned}$$

is a double cover from C to the projective line, showing that C is hyperelliptic. □

Since the function field of every hyperelliptic curve defined over a field k is a quadratic extension of the field $k(x)$, it is possible to give plane models for all hyperelliptic curves.

Theorem 2.2.3. *Every genus g hyperelliptic curves C defined over a field k has a plane model*

$$F(x) = y^2 + h(x)y,$$

where $F(x) = \sum_{i=0}^{2g+2} F_i x^i$, and $h(x), F(x) \in k[x]$ satisfy $\deg(F) \leq 2g+2$ and $\deg(h) \leq g+1$, and the equation $y^2 + h(x)y - F(x)$ is absolutely irreducible.

Proof. See Cassels-Flynn [14] □

Definition 2.2.4. *If $P = (x, y)$ is a point on a hyperelliptic curve C given by a plane model as in Theorem 2.2.3, the point $(x, -h(x) - y)$ also lies on C, we will call this point the* hyperelliptic conjugate *of P and we will denote it by $\overline{P}$.*

Definition 2.2.5. *Let C be a genus g hyperelliptic curve defined over the field k given by a plane model $y^2 + h(x)y = F(x)$ over the field k, as described in Theorem 2.2.3. If* $\mathrm{char}(k) \neq 2$*, we will assume that $h(x) = 0$. If* $\mathrm{char}(k) = 2$ *we will assume that* $\deg(h(x)) \leq g + 1$*, and that $h(x)$ is a monic polynomial.*

1. *We say that this is an* imaginary model *for C if* $\deg(F) = 2g+1$*. When* $\mathrm{char}(k) = 2$ *we also require* $\deg(h) \leq g$*.*

2. *We say that this is a* real model *for C if* $\deg(F) = 2g+2$*. If* $\mathrm{char}(k) = 2$*, then we further require that* $\deg(h) = g + 1$*.*

Lemma 2.2.6. *If the curve C is given by an imaginary model over k, then it will have one k-rational point at infinty. If C is given by a real model, then it will have two points at infinity, possibly defined over a quadratic extension of k.*

Proof. We only prove the lemma for fields k with $\mathrm{char}(k) \neq 2$. Let

$$\phi : C \longrightarrow \mathbf{P}^1,$$
$$(x, y) \mapsto (x : 1).$$

This is a degree 2 endomorphism from a curve of genus g to a curve of genus 0. If e_P denotes the ramification index of ϕ at the point P, the Riemman-Hurwitz formula (see [50][Theorem A.4.2.5]) says that $\sum_{P \in C}(e_P - 1) = 2g + 2$. It can be proved that the affine ramified points are points of the form $(x, 0)$, and that the ramification index at each of these points is 2.

If C is given by a real model, then we have $2g+2$ affine ramification points, and it follows that the points at infinity of C, corresponding to $\phi^*((1 : 0))$, are not ramified, and since the degree of the morphism is 2, there are exactly two

of them.

If C is given by an affine model, then we only have $2g+1$ affine ramification points. It follows that there is at least one ramified point at infinity, which is unique because the degree of ϕ is 2.

Finally, the divisor $\phi^*(1:0)$ is k-rational, proving that the point at infinity is rational when C is given by an imaginary model, and that the points at infinity are defined at most over a quadratic extension of k if C is given by a real model. □

Example 2.2.1. Let C be a hyperelliptic genus 2 curve given by the plane model $C : y^2 = F(x)$, where $F(x) = \sum_{i=0}^{6} a_i x^i$ and $a_6 \neq 0$. Cassels and Flynn give a nonsingular model for C over $\mathbf{P}^4$

$$\begin{aligned} y^2 &= a_0x_0^2 + a_1x_0x_1 + a_2x_1^2 + a_3x_1x_2 + a_4x_2^2 + a_5x_2x_3 + a_6x_3^2, \\ x_0x_2 - x_1^2 &= 0, \\ x_0x_3 - x_1x_2 &= 0, \\ x_1x_3 - x_2^2 &= 0. \end{aligned}$$

The affine points on the plane model of C correspond to the points in the nonsingular model with $x_0 \neq 0$. This correspondence is given by

$$x_i = x^i, y = y.$$

2.2.1 Mumford representation

Definition 2.2.7. *We say that an effective divisor $D = \sum_i P_i$ on a hyperelliptic curve C is* semi-reduced *if $i \neq j$ implies $P_i \neq \overline{P}_j$. If the hyperelliptic curve C has genus g, we say that a divisor D on C is* reduced *if it is semi-reduced,*

and has degree $d \leq g$. We will denote the degree of a divisor D_i as d_i.

Let C be a hyperelliptic curve given by the equation $y^2 + h(x)y = F(x)$. To every pair of polynomials $(u(x), v(x))$ such that

$$u(x) \quad \text{divides} \quad F(x) - h(x)v(x) - v(x)^2, \tag{2.2.1}$$

we associate a divisor as follows

$$\text{If } u(x) = \prod_i (x - r_i), \text{ then } (u(x), v(x)) \mapsto \sum_i (r_i, v(r_i)).$$

Condition (2.2.1) guarantees that the point $(r_i, v(r_i))$ lies on C. Note that the divisor associated to the pair $(u(x), v(x))$ is always an effective affine semi-reduced divisor.

We have seen how to associate an effective affine semi-reduced divisor to every pair of polynomials satisfying condition (2.2.1). Conversely, if $D = \sum_{i=1}^{n} P_i$ is an effective, affine, semi-reduced divisor, and the point P_i has coordinates $P_i = (x_i, y_i)$, let $u(x) = \prod_{i=1}^{n}(x - x_i)$, and let $v(x)$ be the unique polynomial of degree at most $n-1$ passing through the points P_i (if a point P appears m times in the divisor D, we require that $v(x)$ intersects C to order m at P). By construction, the pair of polynomials $(u(x), v(x))$ satisfies condition (2.2.1), and the divisor D is the divisor associated to $(u(x), v(x))$.

Definition 2.2.8. *In the notation of the previous paragraph, we say that $(u(x), v(x))$ is a Mumford representation for the divisor D, and we will denote this as $D = \text{div}(u(x), v(x))$.*

Example 2.2.2. Let C be the curve defined over $\mathbf{Q}$ given by the equation $y^2 = x^5 - 14x^4 + 65x^3 - 112x^2 + 60x$. The pair of polynomials $(x^2 - 301/36x +$

$169/12, 287/216x - 455/72)$ satisfy condition (2.2.1), so they define a divisor D on C. The polynomial $x^2 - 301/36x + 169/12$ is irreducible over $\mathbf{Q}$, so the individual points on the support of D are not $\mathbf{Q}$-rational. The fact that both polynomials in the Mumford representation of D are $\mathbf{Q}$-rational proves that D is however, a $\mathbf{Q}$-rational divisor.

Proposition 2.2.9. *Let C be a hyperelliptic curve defined over the field k. Let D_∞ be a k-rational degree g divisor, and let $[D] \in \mathrm{Cl}^0(C)$ be a k-rational divisor class on the hyperelliptic curve C. Then $[D]$ has a unique representative in $\mathrm{Cl}^0(C)$ of the form $[D_0 - D_\infty]$, where D_0 is an effective k-rational divisor of degree g whose affine part is reduced.*

Proof. The case $D_\infty = g\infty^+$ is Proposition 4.1 of [79]. To prove uniqueness, suppose that D_1 and D_2 are two effective degree g divisors with affine reduced support, and $D_1 - D_\infty \equiv D_2 - D_\infty$. Adding $D_\infty - g\infty^+$ to both sides gives $D_1 - g\infty^+ \equiv D_2 - g\infty^+$. Proposition 4.1 from [79] implies that $D_1 = D_2$.

The existence of the divisor D_0 was proved in Example 2.1.2. □

Definition 2.2.10. *Given a hyperelliptic curve C, we define its* base divisor *D_∞ as:*

- *If C has a unique point at infinity ∞, then $D_\infty = g\infty$.*
- *If C has two points at infinity ∞^+ and ∞^- and even genus, then $D_\infty = \frac{g}{2}(\infty^+ + \infty^-)$.*
- *If C has two points at infinity ∞^+ and ∞^- and odd genus,then $D_\infty = \frac{g+1}{2}\infty^+ + \frac{g-1}{2}\infty^-$. In this case we will further assume that ∞^+ and ∞^- are K-rational points.*

A small problem from a computational point of view is that Proposition 2.2.9 does not guarantee that the supports of D_0 and D_∞ are disjoint, and indeed, in some cases they will have points in common which should be "cancelled out". However, divisors of the form $D_0 - D_\infty$ with D_0 and D_∞ having disjoint support are generic, so it is enough to describe their arithmetic for many applications. In Chapter 3 we will give a complete addition algorithm for hyperelliptic curves; this algorithm becomes very efficient in the generic case.

2.2.2 Points at infinity

Lemma 2.2.11. *Let C be a hyperelliptic curve given by a real model as described in Definition 2.2.5. Denote the two points at infinity on C as ∞^+ and ∞^-. Then the function y/x^{g+1} is well defined and not zero at each of ∞^+ and ∞^-. Furthermore*

$$\frac{y}{x^{g+1}}(\infty^+) \neq \frac{y}{x^{g+1}}(\infty^-).$$

Proof. We have that $\mathrm{ord}_{\infty^+}(x) = \mathrm{ord}_{\infty^-}(x) = -1$ and $\mathrm{ord}_{\infty^+}(y) = \mathrm{ord}_{\infty^-}(y) = -g-1$. Hence $\mathrm{ord}_{\infty^+}(y/x^{g+1}) = \mathrm{ord}_{\infty^-}(y/x^{g+1}) = 0$, proving that the function y/x^{g+1} is well-defined and not zero at each of ∞^+ and ∞^-.

To prove the second part of the lemma, define the function g on C as

$$g(P) = \frac{y(P)}{(y-h)(\overline{P})}.$$

Since $y(P) = (h-y)(\overline{P})$ for every point P, the function g is constant, $g(P) = -1$ for every P. If $\mathrm{char}(k) \neq 2$, then we have that $h(x) = 0$, and since

$x(P) = x(\overline{P})$ and $\infty^+ = \overline{\infty^-}$, it follows that

$$(y/x^{g+1})(\infty^+) = -(y/x^{g+1})(\overline{\infty^-}).$$

If $\mathrm{char}(k) = 2$, then $h(x)$ is monic of degree $g+1$, and we get

$$(y/x^{g+1})(\infty^+) = ((y+h)/x^{g+1})(\infty^-),$$

but the leading term of h is x^{g+1}, so $(h/x^{g+1})(\infty^-) = 1$, in this case

$$(y/x^{g+1})(\infty^+) = (y/x^{g+1})(\infty^-) + 1,$$

and the lemma follows. □

Using Lemma 2.2.11 define

$$a_+ = (y/x^{g+1})(\infty^+), \quad a_- = (y/x^{g+1})(\infty^-),$$

it follows that $a_+ \neq a_-$. Hence, for $p(x)$ a polynomial of the form $p(x) = (a_+x^{g+1} + \sum_{0\leq i\leq g} b_i x^i)$, the function $y - p(x)$ will have valuation strictly larger than $-(g+1)$ at ∞^+ and valuation $-(g+1)$ at ∞^-.

Definition 2.2.12. *In the notation of the previous paragraph, among all degree $g+1$ polynomials $p(x)$ with leading coefficient a_+, there is a unique polynomial in $\overline{k}[x]$ for which the valuation of the function $y - p(x)$ at ∞^+ is maximal; we will denote this polynomial by H^+. Define the polynomial H^- analogously.*

If $C(x, y)$ is the equation of the curve, then $H^+(x)$ and $H^-(x)$ are the polynomials with leading coefficient a_+ and a_- such that $C(x, H^{\pm}(x))$ has

minimal degree. Their coefficients can thus be found recursively. The polynomials $H^{\pm}(x)$ are just a technical tool to specify a point at infinity, similar to the choice of sign when computing the square root of a complex number. Note that the polynomials $H^{\pm}$ are defined over k if and only if the points ∞^+ and ∞^- are k-rational.

Example 2.2.3. Take the hyperelliptic curve C defined over $\mathbf{Q}$ given by equation $y^2 = x^6+4x^5+10x^4+20x^3+30x^2+19x+16$. It is easy to show that $a_+ = 1$ and $a_- = -1$. In this case, $H^+(x) = x^3+2x^2+3x+4$, and $H^-(x) = -H^+(x)$.

Definition 2.2.13. *Given two divisors D_1 and D_2, we will denote the set of pairs of integers (ω^+, ω^-) such that*

$$D_1 \equiv D_2 + \omega^+\infty^+ + \omega^-\infty^-,$$

as $\omega(D_1, D_2)$. We say that the numbers ω^+ and ω^- are counterweights *for D_1 and D_2 if $(\omega^+, \omega^-) \in \omega(D_1, D_2)$.*

The set $\omega(D_1, D_2)$ may be empty. If $[\infty^+ - \infty^-]$ is a torsion point on $\mathrm{Cl}^0(C)$, and the set $\omega(D_1, D_2)$ is not empty, then it is infinite; however this will not affect our algorithms. Given two divisors D_1 and D_2, calculating the values of the counterweights relating them is a difficult problem. When these values are needed in our algorithms, there will be a simple way to calculate them.

2.3 Pairings

Let C be a curve defined over a field k. In this section we describe the Tate- and Weil-pairings on $\mathrm{Cl}^0(C)$ and present Miller's algorithm to compute them.

We conclude with some techniques developed to compute pairings using short Miller loops.

2.3.1 Pairing definition

Let $[D_1] \in \mathrm{Cl}_k^0(C)[r]$ and $D_2 \in \mathrm{Cl}_k^0(C)$ be two divisors with disjoint support. Since rD_1 is principal, there is a function f_{r,D_1} defined over k such that $\mathrm{div}(f_{r,D_1}) = rD_1$. The Tate-pairing is defined as

$$\langle D_1, D_2 \rangle_r = f_{r,D_1}(D_2).$$

Theorem 2.3.1. *The Tate-pairing*

$$\langle \cdot, \cdot \rangle : \mathrm{Cl}_k^0(C)[r] \times \mathrm{Cl}_k^0(C)/r\,\mathrm{Cl}_k^0(C) \longrightarrow k^*/(k^*)^r,$$

is a non-degenerate, Galois-equivariant, bilinear pairing.

Proof. Galois-equivariance follows from the definition of the Tate-pairing. We will first prove that the Tate-pairing is well defined in divisor classes. To prove this for the first entry, let $D_1' = D_1 + \mathrm{div}(g_1)$. Then $f_{r,D_1'} = f_{r,D} \cdot g_1^r$.

To prove that the pairing is well defined on divisor classes on the right, let $D_2' = D_2 + \mathrm{div}(g_2)$. Then

$$\begin{aligned} f_{r,D_1}(D_2') &= f_{r,D_1}(D_2 + \mathrm{div}(g_2)) \\ &= f_{r,D_1}(D_2) \cdot f_{r,D_1}(\mathrm{div}(g_2)) \\ &= f_{r,D_1}(D_2) \cdot g_2(D_1)^r, \end{aligned}$$

where the last equality is given by Weil-reciprocity.

To prove linearity on the left, let $[D_1], [D_2] \in \mathrm{Cl}_k^0(C)[r]$, and $[D_3] \in \mathrm{Cl}_k^0(C)$. Let $[D_{1,2}] = [D_1] + [D_2]$. We have

$$\begin{aligned}\frac{\langle D_1, D_3\rangle_r \langle D_2, D_3\rangle_r}{\langle D_{1,2}, D_3\rangle_r} &= \frac{f_{r,D_1}(D_3) \cdot f_{r,D_2}(D_3)}{f_{r,D_{1,2}}(D_3)} \\ &= g_{D_1,D_2}(D_3)^r,\end{aligned}$$

where g_{D_1,D_2} is a function with associated divisor $D_1 + D_2 - D_{1,2}$.

To prove linearity on the right, let $[D_3] \in \mathrm{Cl}_k^0(C)[r]$ and $[D_1], [D_2], [D_{1,2}] \in \mathrm{Cl}_k^0(C)$ such that $D_1 + D_2 - D_{1,2}$ is the divisor of the function g_{D_1,D_2}. By definition

$$\begin{aligned}\frac{\langle D_3, D_1\rangle_r \langle D_3, D_2\rangle_r}{\langle D_3, D_{1,2}\rangle_r} &= \frac{f_{r,D_3}(D_1) f_{r,D_3}(D_2)}{f_{r,D_3}(D_{1,2})} \\ &= f_{r,D_3}(D_1 + D_2 - D_{1,2}) \\ &= g_{D_1,D_2}(D_3)^r\end{aligned}$$

where the last equality is given by Weil-reciprocity.

We will not prove non-degeneracy of the Tate-pairing. A prove of this fact can be found in [32]

□

If the field k is finite, and contains the group of rth roots of unity μ_r, then it is possible to obtain a unique pairing result simply by defining

$$e(D_1, D_2) = \langle D_1, D_2\rangle_r^{(|k|-1)/r}.$$

This bilinear pairing is known as the *reduced Tate-pairing*.

Definition 2.3.2. *If the curve C is defined over the finite field $\mathbf{F}_q$, and its class group $\mathrm{Cl}^0_{\mathbf{F}_q}(C)$ has an element of prime order r, we define the* embedding degree *of C with respect to r as the smallest positive integer k such that the field $\mathbf{F}_{q^k}$ contains the group of rth roots of unity. In other words, k is the smallest positive integer such that $r \mid (q^k - 1)$. If the embedding degree k is small, we say that C is a* pairing friendly curve.

Definition 2.3.3. *Let $D_1, D_2 \in \mathrm{Cl}^0(C)[r]$ be two r-torsion divisors with disjoint supports. We define the* Weil-pairing *of D_1 and D_2 as*

$$\tilde{e}(D_1, D_2) = \langle D_2, D_1 \rangle_r / \langle D_1, D_2 \rangle_r.$$

Remark 2.3.4. There are alternative definitions of the Weil pairing. See for example Section III.8 in Silverman [91].

Theorem 2.3.5. *The Weil-pairing is a Galois-equivariant, non-degenerate, bilinear pairing*

$$e : \mathrm{Cl}^0(C)[r] \times \mathrm{Cl}^0(C)[r] \longrightarrow \mu_r,$$

where μ_r is the group of rth roots of unity.

Proof. We first prove that the image of the Weil-pairing lies in μ_r. We have that

$$e(D_1, D_2)^r = f_{r,D_2}(rD_1)/f_{r,D_1}(rD_2),$$

since $\mathrm{div}(f_{r,D_1}) = rD_1$ and $\mathrm{div}(f_{r,D_2}) = rD_2$, Weil-reciprocity implies that $f_{r,D_1}(rD_2) = f_{r,D_2}(rD_1)$; hence $e(D_1, D_2)^r = 1$, so $e(D_1, D_2) \in \mu_r$. Galois-equivariance and bilinearity follow from the analogous properties for the Tate-pairing. Again, we do not prove non-degeneracy. A proof can be found in [91][Section III.8]. □

2.3.2 Miller's algorithm

Definition 2.3.6. *Let C be a curve for which there exists a way to select a canonical representative for every element of* $\mathrm{Cl}^0(C)$*. Given a degree 0 divisor D on C and an integer n, let D_n be the canonical representative of the class $[nD]$. We will denote the unique function (up to scalar multiples) with associated divisor $nD - D_n$ as $f_{n,D}$.*

The function $f_{n,D}$ is usually chosen to be normalised at infinity (i.e., that the leading coefficient with respect to a fixed uniformizer at infinity is 1; see [49]). Rather than making such a strong restriction we will just assume that our functions $f_{n,D}$ are defined over the same field as D. Proposition 2.2.9 proves that there is always such a unique representative in the class group of a hyperelliptic curve.

By definition, given two degree 0 divisors D_1 and D_2 on C, if D_3 is the canonical representative of $[D_1 + D_2]$, there is a function whose associated divisor is $D_1 + D_2 - D_3$. Denote this function as g_{D_1,D_2}. Miller's fundamental observation is that

$$f_{n_1+n_2,D} = f_{n_1,D} \cdot f_{n_2,D} \cdot g_{n_1D,n_2D}, \tag{2.3.1}$$

which allows us to compute $f_{r,D}$ (and hence the Tate-pairing) using a square and multiply calculation with $O(\log r)$ steps. Miller's idea idea is presented in Algorithm 2.1.

In several cryptographic applications, a bilinear pairing e and a divisor D are needed such that $e(D, D) \neq 1$. This is typically not possible if e is the Weil- or Tate-pairing and D is defined over the base field of the curve. The following definition tries to circumvent these limitations.

Algorithm 2.1 Miller's Algorithm

INPUT: Divisors D_1 such that $[D_1] \in \mathrm{Cl}^0(C)[r]$, and $D_2 \in \mathrm{Cl}^0(C)$ with disjoint support.
OUTPUT: $f_{r,D_1}(D_2)$.
1: Let $r = 2^n + \sum_{i=0}^{n-1} a_i 2^i$, where $a_i \in \{0,1\}$.
2: Set $v = 1$, $D_3 = D_1$.
3: **for** $1 \leq i \leq n$ **do**
4: $\quad v := v^2 \cdot g_{D_3,D_3}(D_2)$.
5: $\quad D_3 := 2 \cdot D_3$.
6: $\quad$ **if** $a_{n-i} = 1$ **then**
7: $\quad\quad v := v \cdot g_{D_3,D_1}(D_2)$.
8: $\quad\quad D_3 := D_3 + D_1$.
9: **return** v.

Definition 2.3.7. *Let* $e : \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mathbf{G}_T$ *be a non-degenerate bilinear pairing. A morphism* $\psi : \mathbf{G}_1 \longrightarrow \mathbf{G}_2$ *is called a* distortion map *for* $D_1 \in \mathbf{G}_1$ *if* $e(D_1, \psi(D_1)) \neq 1$.

2.3.3 Elliptic Ate and twisted Ate pairing

Several techniques to compute pairings using short Miller loops have been developed. Here we present results obtained by Hess, Smart and Vercauteren in [49]. Let E be an elliptic curve defined over the finite field $\mathbf{F}_q$, with $\#E(\mathbf{F}_q) = q - t + 1$, where t is the trace of Frobenius. Let r be a prime that divides $\#E(\mathbf{F}_q)$, denote $T = t - 1$ and

$$\mathbf{G}_1 = E[r] \cap \mathrm{Ker}(\pi_q - \mathrm{id}), \text{ and } \mathbf{G}_2 = E[r] \cap \mathrm{Ker}(\pi_q - q).$$

Let k denote the embedding degree, define $N = \gcd(T^k - 1, q^k - 1)$ and let $T^k - 1 = LN$.

Theorem 2.3.8 (Theorem 1 of [49])**.** *Given two points* $P \in \mathbf{G}_1$ *and* $Q \in \mathbf{G}_2$*, the function* $e(Q, P) = f_{T,Q}(P)^{(q^k-1)/N}$ *defines a bilinear pairing. If* r *does not*

divide L, then this pairing is non-degenerate. We call it the Ate pairing.

Remark 2.3.9. The Hasse-Weil Theorem says that the number of points of an elliptic curve E defined over the field $\mathbf{F}_q$ is an element of the interval $[q - 2\sqrt{q} + 1, q + 2\sqrt{q} + 1]$. For cryptographic applications, it is customary to choose a curve whose order has a prime divisor r *close* to q (that is, such that $\#E/r$ is small). Note that in this case, the trace of Frobenius t is bounded by $2\sqrt{q}$, so T has approximately half the bit-length of r. In this case, computing the function $f_{T,Q}$ using Miller's algorithm sould be twice as fast as calculating $f_{r,Q}$. There are families of pairing friendly curves where the trace of Frobenius is very small, see for example [37].

Example 2.3.1. We present a construction of pairing-friendly curves due to Barreto and Naehrig, presented in [4]. Define the polynomials

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

Barreto and Naehrig prove that if x_0 is an integer such that $p_0 := p(x_0)$ and $n_0 := n(x_0)$ are both prime numbers, then there is an elliptic curve E defined over the field $\mathbf{F}_{p_0}$ with n_0 points and trace of Frobenius $t(x_0)$. Furthermore, the curve E has embedding degree 12 with respect to n_0 and an equation of the form $y^2 = x^3 + b$, where b can be easily computed.

In [49, Section 6], Hess, Smart and Vercauteren prove that if E has a twist of degree d, embedding degree k, and we set $m = \gcd(d, k)$ and $e = k/m$, then

Theorem 2.3.10. *In the notation of Theorem 2.3.8, the function*

$$e(P,Q) = f_{T^e,P}(Q)^{(q^k-1)/r}, \tag{2.3.2}$$

defines a bilinear function on $\mathbf{G}_1 \times \mathbf{G}_2$*, called the twisted Ate pairing.*

2.3.4 Hyperelliptic Ate pairings

We have seen that in some cases it is possible to compute pairings using a function $f_{n,D}$ where n is much smaller than required for the Tate-pairing. We will revisit some of these techniques in the case of hyperelliptic curves.

Let C be a hyperelliptic curve defined over a finite field $\mathbf{F}_q$. Denote the Frobenius automorphism of C as π_q. Let

$$\mathbf{G}_1 = \mathrm{Cl}^0_{\mathbf{F}_{q^k}}(C)[r] \cap \mathrm{Ker}(\pi_q - \mathrm{id}) \text{ and } \mathbf{G}_2 = \mathrm{Cl}^0_{\mathbf{F}_{q^k}}(C)[r] \cap \mathrm{Ker}(\pi_q - q),$$

denote the 1- and q-eigenspaces of π_q in the r-torsion subgroup of $\mathrm{Cl}^0_{\mathbf{F}_{q^k}}(C)$. If $D_1 \in \mathbf{G}_1$ and $D_2 \in \mathbf{G}_2$ are divisors on C, the authors of [46] proved:

Theorem 2.3.11. *The function* $e_q : \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mu_r$*, given by*

$$e_q(D_1, D_2) = f_{q,D_1}(D_2)^{(q^k-1)/r},$$

defines a non-degenerate bilinear pairing on $\mathbf{G}_1 \times \mathbf{G}_2$*.*

2.3.5 R-ate pairings

Let $\mathbf{G}_1$ and $\mathbf{G}_2$ be subgroups of the class group of a curve C (please note that $\mathbf{G}_1$ and $\mathbf{G}_2$ needn't be Frobenius eigenspaces in this section). If $D_1 \in \mathbf{G}_1$ and

$D_2 \in \mathbf{G}_2$, Lee, Lee and Park prove in [64] the following:

Theorem 2.3.12. *[Theorem 3.2 in [64]] Let A, B, a, b be integers such that $A = aB+b$, where the functions $f_{A,D}$ and $f_{B,D}$ define bilinear maps in $\mathbf{G}_1 \times \mathbf{G}_2$. Then the function*

$$f_{a,BD}(E) \cdot f_{b,D}(E) \cdot g_{aBD,bD}(E),$$

defines a bilinear map in $\mathbf{G}_1 \times \mathbf{G}_2$.

Remark 2.3.13. Note that if B is the order of D, then the functions $f_{a,BD}$ and $g_{aBD,bD}$ are constant, so the function $f_{b,D}(E)$ will define a bilinear map.

Chapter 3

Arithmetic on Hyperelliptic Curves

In this chapter we will present addition algorithms for the class group of a hyperelliptic curve. Firstly, we will present Cantor's algorithm; this is a classic addition algorithm for curves given by an imaginary model. We will then describe our proposal for addition on curves given by a real model and compare it with previous proposals.

Algorithms analogous to the ones presented in this chapter were developed by Mike Harrison and implemented as part of the computer algebra system Magma. The algorithms corresponding to the results in this chapter were first released in Magma V2.12 in July 2005. Our research was done independently. The results of this chapter are published jointly with Mike Harrison in the article [36].

Proposition 2.2.9 shows that for a genus g hyperelliptic curve C, given any degree g divisor D_∞, every divisor class has a unique representative of the form $D - D_\infty$, where D is an effective divisor with reduced affine part. The

problem of finding explicit addition algorithms on $\mathrm{Cl}^0(C)$ can be restated as:

Problem 3.1. Let C be a genus g hyperelliptic curve with a fixed degree g divisor D_∞ on C. Given two effective degree g divisors D_1 and D_2 with reduced affine part, find an effective degree g divisor D_3 with reduced affine part such that

$$[D_1 - D_\infty] + [D_2 - D_\infty] = [D_3 - D_\infty].$$

Under this perspective, finding an efficient addition algorithm on the class group of a hyperelliptic curve is a problem in two stages. First, one needs to find an appropriate divisor D_∞, and then the algorithms to compute with this base divisor need to be developed. In this chapter we prove that using a divisor D_∞ which is more balanced at infinity, arithmetic in the class group may be performed more efficiently than done by [26, 55, 79, 80]. In the case of genus 2 curves, all explicit addition formulae presented so far [26] can be used with our representation, giving improved results (see Table 3.1).

3.1 Arithmetic on Curves Given by Imaginary Models

Let C be a degree g hyperelliptic curve defined over the field k by the equation $C : y^2 + h(x)y = F(x)$, where F is a polynomial of degree $2g+1$. If we denote the unique point at infinity of C as ∞, and let $D_\infty = g\infty$, Proposition 2.2.9 says that every divisor class on $\mathrm{Cl}^0(C)$ has a unique representative of the form $D - D_\infty$, where D is an effective divisor with reduced affine part. This can also be interpreted as follows

Proposition 3.1.1. *With C a curve as described above, there is a natural*

bijection between $\mathrm{Cl}^0(C)$ *and the set of reduced affine divisors* $\mathrm{RedDiv}(C)$ *on* C*. This bijection is given by*

$$\begin{aligned} \mathrm{div}_C : \mathrm{RedDiv}(C) &\longrightarrow \mathrm{Cl}^0(C) \\ D &\mapsto [D - d\infty], \end{aligned}$$

where $d = \deg(D)$*.*

This proposition allows us to restrict to reduced affine divisors in order to describe an addition algorithm in $\mathrm{Cl}^0(C)$. Since every such divisor can be given by its Mumford representation, our algorithms will operate on Mumford representations.

Algorithm 3.1 Composition

INPUT: Semi-reduced affine divisors $D_1 = \mathrm{div}(u_1, v_1)$ and $D_2 = \mathrm{div}(u_2, v_2)$.
OUTPUT: A semi-reduced affine divisor $D_3 = \mathrm{div}(u_3, v_3)$.

1: Compute s (monic), $f_1, f_2, f_3 \in k[x]$ such that

$$s = \gcd(u_1, u_2, v_1 + v_2 + h) = f_1 u_1 + f_2 u_2 + f_3(v_1 + v_2 + h).$$

2: Set $u_3 := u_1 u_2 / s^2$ and $v_3 := \left(f_1 u_1 v_2 + f_2 u_2 v_1 + f_3(v_1 v_2 + F)\right)/s \mod u_3$
3: **return** $\mathrm{div}(u_3, v_3)$.

The result D_3 of Algorithm 3.1 will be denoted $D_3 = \mathrm{comp}(D_1, D_2)$. The divisor of the function s from Algorithm 3.1 is

$$\mathrm{div}(s) = D_1 + D_2 - D_3 - (d_1 + d_2 - d_3)(\infty), \tag{3.1.1}$$

which proves that

$$[D_1 - d_1\infty] + [D_2 - d_2\infty] = [D_3 - d_3\infty]$$

Algorithm 3.1 is also known as *divisor composition.*

Example 3.1.1. Let C be a hyperelliptic curve defined over the field $\mathbf{F}_{101}$ given by the equation $y^2 + (x + 12)y = x^7 + 43x + 19$. Let $D_1 = \mathrm{div}(x^3 + 85x^2+23x+49, 64x^2+100x+9)$ and $D_2 = (x^3+79x^2+96x+73, 25x^2+70x+55)$. Then

$$D_3 = \mathrm{comp}(D_1, D_2) = \mathrm{div}(x^4+85x^3+99x^2+73x+93, 94x^3+24x^2+93x+86).$$

Note that D_3 has degree 4 because D_1 has the point $(11, 66)$ in its divisor of zeros, and D_2 has its hyperelliptic conjugate $(11, 12)$. In this case the polynomial $s(x)$ from Algorithm 3.1 is $s(x) = x - 11$, and $(f_1, f_2, f_3) = (93, 8, 97)$.

Given an affine semi-reduced divisor D_0 of degree $d_0 \geq g+1$, Algorithm 3.2 finds another affine semi-reduced divisor D_1 with smaller degree d_1, such that

$$[D_0 - d_0\infty] = [D_1 - d_1\infty] \tag{3.1.2}$$

Algorithm 3.2 is known as *divisor reduction.*

Algorithm 3.2 Reduction

INPUT: A semi-reduced affine divisor $D_0 = \mathrm{div}(u_0, v_0)$, with $d_0 \geq g + 1$.
OUTPUT: A semi-reduced affine divisor $D_1 = \mathrm{div}(u_1, v_1)$ such that Equation (3.1.2) holds.
1: Set $u_1 := (v_0^2 + hv_0 - F)/u_0$ made monic.
2: Let $v_1 := (-v_0 - h) \mod u_1$.
3: **return** $\mathrm{div}(u_1, v_1)$.

The result D_1 of Algorithm 3.2 will be denoted as $D_1 = \mathrm{red}(D_0)$. The geometric interpretation of Algorithm 3.2 is very simple: given the effective affine divisor $D_0 = \mathrm{div}(u_0, v_0)$, we know (by definition of the Mumford representation) that the divisor of zeros D_z of the function $y - v_0(x)$ has (in the

notation of Algorithm 3.2) $D_z = D_0 + \overline{D}_1$, and if $\deg(u_0) \geq g+1$, then the degree of D_z satisfies $\deg(D_z) < 2\deg(D_0)$, hence $\deg(D_1) < \deg(D_0)$, and we have

$$\operatorname{div}\left(\frac{y - v_0(x)}{u_0}\right) = D_0 - D_1 - (d_0 - d_1)(\infty). \tag{3.1.3}$$

It follows that

$$[D_0 - d_0\infty] = [D_1 - d_1\infty].$$

Example 3.1.2. If we continue with Example 3.1.1, and we compute $D_4 = \operatorname{red}(D_3)$, we get

$$D_4 = \operatorname{div}(x^3 + 68x^2 + 12x + 79, 5x^2 + 24x + 56).$$

Algorithm 3.3 Cantor's Algorithm

INPUT: Reduced affine divisors $D_1 = \operatorname{div}(u_1, v_1)$ and $D_2 = \operatorname{div}(u_2, v_2)$.
OUTPUT: A reduced affine divisor $D_3 = \operatorname{div}(u_3, v_3)$ such that $[D_1 - d_1\infty] + [D_2 - d_2\infty] = [D_3 - d_3\infty]$.

1: Let $D := \operatorname{comp}(D_1, D_2)$.
2: **while** $\deg(D) > g$ **do**
3: $\quad D := \operatorname{red}(D)$.
4: **return** D.

Example 3.1.3. Using Examples 3.1.1 and 3.1.2, it follows that if $D_1 = \operatorname{div}(x^3 + 85x^2 + 23x + 49, 64x^2 + 100x + 9)$ and $D_2 = (x^3 + 79x^2 + 96x + 73, 25x^2 + 70x + 55)$ are two divisors on the curve C. Then the divisor $D_4 = \operatorname{div}(x^3 + 68x^2 + 12x + 79, 5x^2 + 24x + 56)$ is such that

$$[D_1 - 3\infty] + [D_2 - 3\infty] = [D_4 - 3\infty].$$

3.2 Algorithms on Real Models

In this section we modify the algorithms presented in the last section to compute with divisor classes of hyperelliptic curves. We will analyse these algorithms as operations on the Mumford representation of an affine semi-reduced divisor. Our main contribution is to give a geometric interpretation of these algorithms.

Algorithm 3.4 Modified Composition

INPUT: Semi-reduced affine divisors $D_1 = \mathrm{div}(u_1, v_1)$ and $D_2 = \mathrm{div}(u_2, v_2)$.
OUTPUT: A semi-reduced affine divisor $D_3 = \mathrm{div}(u_3, v_3)$ and a pair (ω^+, ω^-), such that $(\omega^+, \omega^-) \in \omega(D_1 + D_2, D_3)$.

1: Compute s (monic), $f_1, f_2, f_3 \in k[x]$ such that

$$s = \gcd(u_1, u_2, v_1 + v_2 + h) = f_1 u_1 + f_2 u_2 + f_3(v_1 + v_2 + h).$$

2: Set $u_3 := u_1 u_2 / s^2$ and $v_3 := \left(f_1 u_1 v_2 + f_2 u_2 v_1 + f_3(v_1 v_2 + F)\right)/s \mod u_3$
3: **return** $\mathrm{div}(u_3, v_3)$ and $(\deg(s), \deg(s))$.

The result D_3 of Algorithm 3.4 will be denoted $D_3, (\omega^+, \omega^-) = \mathrm{comp}(D_1, D_2)$. The divisor of the function s from Algorithm 3.4 is

$$\mathrm{div}(s) = D_1 + D_2 - D_3 - \frac{d_1 + d_2 - d_3}{2}(\infty^+ + \infty^-), \tag{3.2.1}$$

which proves that

$$(\omega^+, \omega^-) \in \omega(D_1 + D_2, D_3).$$

Algorithm 3.4 is also known as *divisor composition.*

Example 3.2.1. Let C be a genus 3 hyperelliptic curve defined over the field $\mathbf{F}_{127}$ by the equation $y^2 = x^8 + 2x^5 + x^4 + 4x^2 + 88x + 45$. Let D_1 and D_2 be

divisors given by

$$D_1 = \mathrm{div}(x^3 + 35x^2 + 47x + 51, 68x^2 + x + 41),$$
$$D_2 = \mathrm{div}(x^2 + 121x + 100, 37x + 113).$$

The result $D_3, (\omega^+, \omega^-) = \mathrm{comp}(D_1, D_2)$ of the composition of D_1 and D_2 is given by $(\omega^+, \omega^-) = (0, 0)$ and

$$D_3 = \mathrm{div}(x^5 + 29x^4 + 64x^3 + 94x^2 + 76x + 20, 6x^4 + 40x^3 + 115x^2 + 64x + 7).$$

Given an affine semi-reduced divisor D_0, of degree $d_0 \geq g+2$, Algorithm 3.5 finds another affine semi-reduced divisor D_1 with smaller degree d_1, and a pair of integers (ω^+, ω^-) such that

$$(\omega^+, \omega^-) \in \omega(D_0, D_1) \tag{3.2.2}$$

Algorithm 3.5 is a modification of the divisor reduction algorithm (Algorithm 3.2), the only difference is that Algorithm 3.5 also returns a pair of counterweights.

The result D_1 of Algorithm 3.5 will be denoted as $D_1, (\omega^+, \omega^-) = \mathrm{red}(D_0)$. The geometric interpretation of Algorithm 3.5 is very simple: given the effective affine divisor $D_0 = \mathrm{div}(u_0, v_0)$, we know (by definition of the Mumford representation) that the divisor of zeros D_z of the function $y - v_0(x)$ has (in the notation of Algorithm 3.5) $D_z = D_0 + \overline{D}_1$, and if $\deg(u_0) \geq g+2$, then the degree of D_z satisfies $\deg(D_z) < 2\deg(D_0)$, hence $\deg(D_1) < \deg(D_0)$, and if

Algorithm 3.5 Modified Reduction

INPUT: A semi-reduced affine divisor $D_0 = \mathrm{div}(u_0, v_0)$, with $d_0 \geq g + 2$.

OUTPUT: A semi-reduced affine divisor $D_1 = \mathrm{div}(u_1, v_1)$ and a pair (ω^+, ω^-), such that $d_1 < d_0$ and Equation (3.2.2) holds.

1: Set $u_1 := (v_0^2 + hv_0 - F)/u_0$ made monic.
2: Let $v_1 := (-v_0 - h) \mod u_1$.
3: **if** the leading term of v_0 is $a_+ x^{g+1}$ (in the notation of Definition 2.2.12) **then**
4: Let $(\omega^+, \omega^-) := (d_0 - g - 1, g + 1 - d_1)$.
5: **else if** the leading term of v_0 is $a_- x^{g+1}$ **then**
6: Let $(\omega^+, \omega^-) := (g + 1 - d_1, d_0 - g - 1)$.
7: **else**
8: Let $(\omega^+, \omega^-) := (\frac{d_0 - d_1}{2}, \frac{d_0 - d_1}{2})$.
9: **return** $\mathrm{div}(u_1, v_1), (\omega^+, \omega^-)$.

the leading term of v_0 is different to that of $H^{\pm}$ we have

$$\mathrm{div}\left(\frac{y - v_0(x)}{u_0}\right) = D_0 - D_1 - \frac{d_0 - d_1}{2}(\infty^+ + \infty^-). \tag{3.2.3}$$

It follows that

$$D_0 - D_1 \equiv \frac{d_0 - d_1}{2}(\infty^+ + \infty^-).$$

A similar analysis when the leading coefficient of v_0 coincides with that of $H^{\pm}$ shows that if $D_1, (\omega^+, \omega^-) = \mathrm{red}(D_0)$, then we always have $(\omega^+, \omega^-) \in \omega(D_0, D_1)$.

Example 3.2.2. We will continue with Example 3.2.1. We want to compute $D_4, (\omega^+, \omega^-) = \mathrm{red}(D_3)$. It is straightforward to verify that $(\omega^+, \omega^-) = (1, 1)$ and

$$D_4 = \mathrm{div}(x^3 + 21x^2 + 29x + 45, 31x^2 + 125x + 60).$$

Algorithm 3.6 is only defined for affine semi-reduced divisors on curves given by a real model. If it were applied on a divisor of degree at least $g + 2$, Algorithm 3.6 would coincide with Algorithm 3.5. When applied on a divisor

Algorithm 3.6 Composition at Infinity and Reduction

INPUT: A semi-reduced affine divisor $D_0 = \text{div}(u_0, v_0)$ of degree $d_0 \leq g + 1$.
OUTPUT: A reduced affine divisor $D_1 = \text{div}(u_1, v_1)$ and a pair of integers (ω^+, ω^-) such that $(\omega^+, \omega^-) \in \omega(D_0, D_1)$.

1: $v_1' := H^{\pm} + (v_0 - H^{\pm} \mod u_0)$,
2: $u_1 := (v_1'^2 + hv_1' - F)/u_0$ made monic.
3: $v_1 := -h - v_1' \mod u_1$.
4: **if** H^+ was used **then**
5: Let $(\omega^+, \omega^-) := (d_0 - g - 1, g + 1 - d_1)$.
6: **else if** H^- was used **then**
7: Let $(\omega^+, \omega^-) := (g + 1 - d_1, d_0 - g - 1)$.
8: **return** $\text{div}(u_1, v_1), (\omega^+, \omega^-)$.

D_0 degree at most $g + 1$, Algorithm 3.6 can be interpreted as composing the divisor D_0 with some divisor at infinity, followed by Algorithm 3.5. The polynomial v_1' in this algorithm is the equivalent to polynomial v_3 in Algorithm 3.4. The result D_1 of this algorithm will be denoted as $D_1, (\omega^+, \omega^-) = \text{red}_\infty(D_0)$. Formally, the action of this algorithm is given by the following.

Proposition 3.2.1. *Given an effective semi-reduced divisor with affine support D_0, with Mumford representation* $\text{div}(u_0, v_0)$ *and degree $d_0 \leq g + 1$. If $D_1, (\omega^+, \omega^-) = \text{red}_\infty(D_0)$, then*

$$(\omega^+, \omega^-) \in \omega(D_0, D_1).$$

Proof. We will only prove this when the algorithm is applied using H^+. Notice that the polynomial $v_1'(x)$ has the property that the function $f = y - v_1'(x)$ has all the points in D_0 in its divisor of zeros.

The $(g + 1) - d_0$ highest degree coefficients of $v_1'(x)$ coincide with those of $H^+(x)$, so the function

$$(v_1'(x))^2 + hv_1'(x) - F(x),$$

which finds the affine support of f, has degree at most $g + d_0$, and it follows that the affine support of f has at most $g + d_0$ points.

We know that the function $y - v_1'(x)$ will have valuation $-(g+1)$ at ∞^-. The divisor of f is then:

$$\mathrm{div}(f) = D_0 + D_2 - (d_0 + d_2 - (g+1))\infty^+ - (g+1)\infty^- \tag{3.2.4}$$

If we denote by D_1 the hyperelliptic conjugate of D_2, we know that

$$\mathrm{div}(u_1) = D_2 + D_1 - d_2(\infty^+ + \infty^-)$$

which together with Equation (3.2.4) implies

$$\mathrm{div}\left(\frac{y - v_1'(x)}{u_1}\right) = D_0 - D_1 - (d_0 - (g+1))\infty^+ - (g+1-d_2)\infty^- \tag{3.2.5}$$

which trivially becomes

$$D_0 \equiv D_1 + (d_0 - (g+1))\infty^+ + (g+1-d_1)\infty^-. \tag{3.2.6}$$

The proposition follows at once.

□

Example 3.2.3. We continue with the calculations from Examples 3.2.1 and 3.2.2. We want to compute $D_5, (\omega^+, \omega^-) = \mathrm{red}_\infty(D_4)$. In this case we will use the polynomial $H^+ = x^4 + x + 64$. Again, following Algorithm 3.6, we get $(\omega^+, \omega^-) = (-1, 2)$ and

$$D_5 = \mathrm{div}(x^2 + 8x + 57, 39x + 14).$$

Remark 3.2.2. When dealing with explicit computations, the divisors D_0 and D_1 will very often have degree g, in which case we can re-write Equation (3.2.6) as

$$D_0 + (\infty^+ - \infty^-) \equiv D_1.$$

Choosing any degree g base divisor D_∞ to represent the points on the class group of C, this equation tells us that

$$(D_0 - D_\infty) + (\infty^+ - \infty^-) \equiv (D_1 - D_\infty),$$

in other words, Algorithm 3.6 is nothing but addition of $\infty^+ - \infty^-$; this turns out to be such a simple operation because the divisor composition is elementary and can easily be incorporated in the divisor reduction process, which is itself very simple.

We would like to emphasize that Algorithm 3.6 is independent of the choice of base divisor, so one has the freedom to choose a divisor D_∞ optimal in each specific case.

Remark 3.2.3. Previous authors have used the terms "baby steps" and "giant steps". We explain these using our notation. Given two divisors $D_1 = \mathrm{div}(u_1, v_1)$ and $D_2 = \mathrm{div}(u_2, v_2)$ on C, a *"giant step"* on D_1 and D_2 is the result of computing $D_3 = \mathrm{comp}(D_1, D_2)$ and successively applying reduction steps (using a red_∞ reduction) on the result until the degree of $\mathrm{red}^i_\infty(D_3)$ is at most g. "Baby steps" are only defined on reduced affine effective divisors, and the result of a "baby step" on a reduced divisor D is the divisor $\mathrm{red}_\infty(D)$.

Remark 3.2.4. We have just seen that Algorithm 3.6 generically corresponds to addition of $\infty^+ - \infty^-$, however, it has long been claimed that this operation has no analogue in the imaginary curve case. Using the previous remark, we

propose the following.

Let $C : y^2 = G(x)$, where $\deg(G(x)) = 2g+1$, be a non-singular imaginary model for a hyperelliptic curve of genus g. Take a point $P = (x_P, y_P)$ on C. Given an effective affine divisor $D = \mathrm{div}(u_0, v_0)$ on C, where $\deg(v_0) < \deg(u_0)$, define a P-baby step on D as follows:

$$\begin{aligned} a &= (y_P - v_0(x_P))/u_0(x_P) \\ \tilde{v}_1(x) &= au_0(x) + v_0(x) \\ u_1(x) &= \frac{(\tilde{v}_1)^2 - G(x)}{(x - x_P)u_0(x)} \\ v_1(x) &= -\tilde{v}_1 \mod u_1(x) \end{aligned}$$

The result of applying a P-baby step on the divisor D_0 is, generically, a divisor D_1 such that $D_0 + (P - \infty) = D_1$. This algorithm will fail when P is in the support of D_0. Doing some precomputations and using an appropriate implementation, this operation should be as efficient as Algorithm 3.6. A good choice of P (for instance, having a very small x_P, or even $x_P = 0$) could have a big impact on the efficiency of this algorithm. In the last section of this chapter, Algorithm 3.9 gives explicit formulae to calculate a P-baby step in genus 2 curves, where the point P has the form $P = (0, y)$. The operation count of Algorithm 3.9 is (1I,1S,5M), which is very competitive compared with the (1I,2S,4M) required by its analogue in curves given by a real model [26].

The following technical lemma will be used in the next section to prove that our proposed addition algorithm finishes.

Lemma 3.2.5. *Let D_0 be an effective divisor of degree $d_0 = 2g$ and D_1 be an*

effective affine divisor of degree $d_1 \leq g$ *. If* $(\omega_1^+, \omega_1^-) \in \omega(D_0, D_1)$*,*

$$D_2, (\omega_r^+, \omega_r^-) = \mathrm{red}_\infty(D_1) \quad \textit{(using } H^+\textit{)},$$

and we denote $(\omega_2^+, \omega_2^-) = (\omega_1^+ + \omega_r^+, \omega_1^- + \omega_r^-)$*, then* $(\omega_2^+, \omega_2^-) \in \omega(D_0, D_2)$ *and*

$$\omega_1^+ - \omega_1^- > \omega_2^+ - \omega_2^-.$$

If $\omega_1^- < (g-1)/2$ *then* $\omega_2^+ \leq g/2$*.*

Proof. From the hypotheses we know that $\omega_1^+ + \omega_1^- = 2g - d_1$. Proposition 3.2.1 says that

$$(\omega_r^+, \omega_r^-) = (d_1 - (g+1), g + 1 - d_2), \tag{3.2.7}$$

this implies that

$$\omega_1^+ - \omega_1^- = \omega_2^+ - \omega_2^- + (2g + 2 - d_0 - d_1),$$

which proves the first assertion. Equation (3.2.7) together with $\omega_1^+ = 2g - d_1 - \omega_1^-$ implies

$$\begin{aligned} \omega_2^+ &= \omega_1^+ + d_1 - g - 1 \\ &= (2g - d_1 - \omega_1^-) + d_1 - g - 1 \\ &= g - 1 - \omega_1^- \end{aligned}$$

by hypothesis $\omega_1^- < (g-1)/2$, so that $\omega_2^+ > (g-1)/2$, and since ω_2^+ is an integer, the result follows.

□

Explicit formulae implementing the algorithms presented in this section for the genus 2 case have been given in [26, 55]. All the explicit addition formulae presented so far (specially for $g = 2$) that we have knowledge of (including those of [26, 55]) first compute the composition of the two affine divisors in the summands, then find the divisor with degree at most $g + 1$ which is the result of successively applying reduction steps, and finally give an explicit form of Algorithm 3.6. For example, in [55] an algorithm is given to efficiently compute a giant step. It can then be used in any arithmetic application that requires such an operation, regardless of the representation of divisors in $\mathrm{Cl}^0(C)$ being used. Hence, it is possible to use these formulae to compute divisor addition using our proposal with no alterations.

3.3 Addition on Real Models

Throughout this section C will denote a genus g hyperelliptic curve defined over a field k, given by the equation

$$C : y^2 + h(x)y = F(x),$$

where $F(x)$ is a degree $2g + 2$ polynomial. If $\mathrm{char}(k) \neq 2$, then we will further assume that $h = 0$. If $\mathrm{char}(k) = 2$, then h will be monic and $\deg(h) = g + 1$.

We will also assume that the divisor D_∞ from Definition 2.2.10 is k-rational. This condition holds automatically for even g. For odd values of g one needs to further assume that the leading coefficient of F is a square in k if $\mathrm{char}(k) \neq 2$ or that the leading coefficient of F is of the form $\omega^2 + \omega$ if $\mathrm{char}(k) = 2$.

Definition 3.3.1. *Every element* $[a_0]$ *of* $\mathrm{Cl}^0(C)$ *has a unique representative of*

the form $a_0 = D_0 - D_\infty$, where D_0 is a degree g effective divisor with reduced affine part. Any effective, degree g divisor D_0 can be uniquely written as $D_0 = D_0' + n_0\infty^+ + m_0\infty^-$, where D_0' is the affine support of D_0, and $n_0, m_0 \in \mathbb{Z}_{\leq 0}$; in this case we will denote the divisor $D_0 - D_\infty$ as $\mathrm{div}((u_0, v_0), n_0)$, where $\mathrm{div}(u_0, v_0) = D_0'$ is the Mumford representation of D_0'. This representation of a divisor is unique.

We would like to remark that in the notation we have just described for divisors, we always have $\deg v_0 < \deg u_0$ and n_0 is an integer such that $0 \leq n_0 \leq g - \deg(u_0)$.

Problem 3.2. Given two divisors $a_1 = \mathrm{div}((u_1, v_1), n_1)$ and $a_2 = \mathrm{div}((u_2, v_2), n_2)$ of $\mathrm{Cl}^0(C)$, we want to find $a_3 = \mathrm{div}((u_3, v_3), n_3)$ such that

$$[a_1] + [a_2] = [a_3].$$

To fix notation, let

$$\begin{aligned} a_i &= \mathrm{div}(u_i, v_i) + n_i\infty^+ + m_i\infty^- - D_\infty, \\ \tilde{D}_i &= \mathrm{div}(u_i, v_i) + n_i\infty^+ + m_i\infty^-, \\ D_i &= \mathrm{div}(u_i, v_i) \end{aligned}$$

for $i \in 1, 2$.

Throughout Algorithm 3.7 we always have that $(\omega^+, \omega^-) \in \omega(\tilde{D}_1 + \tilde{D}_2, D)$. We have mentioned that if $\deg(D) \geq g+2$ then $\deg(\mathrm{red}(D)) < \deg(D)$, so step 3 always finishes. Lemma 3.2.5 proves that step 4, and hence the algorithm, always finish.

Finally, the condition ($\omega^+ < g/2$ or $\omega^- < (g-1)/2$) from Step 4 can be

Algorithm 3.7 Divisor Addition

INPUT: Divisors $a_i = \mathrm{div}((u_i, v_i), n_i)$ for $i \in \{1, 2\}$.
OUTPUT: $a_3 = \mathrm{div}((u_3, v_3), n_3), [a_3] = [a_1] + [a_2]$.
1: Set $(\omega^+, \omega^-) := (n_1 + n_2, m_1 + m_2)$.
2: Let $D, (a, b) := \mathrm{comp}(D_1, D_2)$. Update $(\omega^+, \omega^-) := (\omega^+ + a, \omega^- + b)$.
3: **while** $\deg(D) > g + 1$ **do**
4: $\quad D, (a, b) := \mathrm{red}(D)$. Update $(\omega^+, \omega^-) := (\omega^+ + a, \omega^- + b)$.
5: **while** $\omega^+ < g/2$ or $\omega^- < (g-1)/2$ **do**
6: $\quad D, (a, b) := \mathrm{red}_\infty(D)$. Update $(\omega^+, \omega^-) := (\omega^+ + a, \omega^- + b)$.
7: $\quad$ Use H^+ in red_∞ if $\omega^+ > \omega^-$, else use H^-.
8: Let $E := D + \omega^+\infty^+ + \omega^-\infty^- - D_\infty$.
9: Now E is an effective degree g divisor. Write $E = D + n_3\infty^+ + m_3\infty^-$, where D is an effective affine divisor.
10: **return** $\mathrm{div}(D, n_3)$.

easily explained. At every step in the algorithm we have that $\tilde{D}_1 + \tilde{D}_2 \equiv D + \omega^+\infty^+ + \omega^-\infty^-$. Since formally we are interested in elements of $\mathrm{Cl}^0(C)$ (that is, even though in our algorithms we only ever deal with reduced divisors, it is the class in $\mathrm{Cl}^0(C)$ that they represent that interests us), this relation becomes

$$\tilde{D}_1 - D_\infty + \tilde{D}_2 - D_\infty \equiv (D + \omega^+\infty^+ + \omega^-\infty^- - D_\infty) - D_\infty,$$

and the condition from Step 4 guarantees that when D_∞ is given by Definition 2.2.10, the divisor

$$E = D + \omega^+\infty^+ + \omega^-\infty^- - D_\infty,$$

is effective, ensuring that our algorithm returns the canonical representative of the divisor class.

Example 3.3.1. Let C be the hyperellipitc curve defined over $\mathbf{F}_{97}$ given by equation $C : y^2 = x^6 + 13x^2 + 92x + 7$. Consider divisors $D_1 = \mathrm{div}(x^2 + 75x +$

$57, x + 13)$ and $D_2 = (x^2 + 38x + 41, x + 25)$.

$$\begin{aligned}
\text{comp}(D_1, D_2) &= D_3, (0, 0) \\
D_3 &= \text{div}(x^4 + 16x^3 + 38x^2 + 3x + 9, 20x^3 + 2x^2 + 50x + 84), \\
\text{red}(D_3) &= D_4, (1, 1) \\
D_4 &= \text{div}(x^2 + 53x + 81, 10x + 63).
\end{aligned}$$

Putting all this information together, we get

$$D_1 + D_2 \equiv D_4 + \infty^+ + \infty^-,$$

and using $D_\infty = (\infty^+ + \infty^-)$ as base divisor, we get

$$[D_1 - D_\infty] + [D_2 - D_\infty] = [D_4 - D_\infty].$$

Cantor's addition algorithm for curves given by an imaginary model (see [13] or Algorithm 3.3) can be seen as a degenerate case of our algorithm. We can think of Algorithm 3.7 as:

1. Divisor composition.
2. Reduction steps until the degree is at most $g + 1$.
3. Use red_∞ to balance the divisor at infinity.

Since imaginary models have a unique point at infinity, to perform divisor addition it suffices to compute the composition and reduction steps, making the balancing step redundant. In the following section we will argue that our

divisor D_∞ is the correct choice to have an algorithm analogous to that of Cantor.

Remark 3.3.2. If C has even genus, the points ∞^+ and ∞^- are not k-rational and the divisors a_1 and a_2 are k-rational, by a simple rationality argument the counterweights will always be equal, hence the addition algorithm will produce a divisor D with equal counterweights such that $\deg(D) \leq g$ in step 3. Algorithm 3.7 will then finish and step 4 will not be necesary. In this case the (non k-rational) polynomials $H^\pm$ will not be used and no red_∞ step will be computed.

This last observation suggests that, given a hyperelliptic curve C with even genus, one could move two non k-rational points to infinity and get an addition law completely analogous to Cantor's algorithm. This trivial trick could greatly simplify the arithmetic on C.

One key operation in an efficiently computable group is element inversion. Algorithm 3.8 describes this operation in $\text{Cl}^0(C)$.

Algorithm 3.8 Divisor Inversion

INPUT: A divisor $a_1 = \text{div}((u_1, v_1), n_1)$.
OUTPUT: A divisor $a_2 = \text{div}((u_2, v_2), n_2)$ such that $[a_1] = -[a_2]$.
1: **if** g is even **then**
2: **return** $\text{div}((u_1, (-h - v_1 \mod u_1)), g - \deg(u_1) - n_1)$.
3: **else if** g is odd and $n_1 > 0$ **then**
4: **return** $\text{div}((u_1, (-h - v_1 \mod u_1)), g - m_1 - \deg(u_1) + 1)$.
5: **else**
6: Let $D_1 = \text{red}_\infty(\text{div}(u_1, -h - v_1))$.
7: **return** $\text{div}(D_1, 0)$.

Given the geometric analysis that we have made of the addition algorithm, computing pairings on the class group of an arbitrary hyperelliptic curve can be done following Miller's algorithm. An analysis of pairing implementations using the algorithms described in this chapter is presented in Chapter 5.

3.3.1 Other proposals

Previous proposals for addition algorithms on hyperelliptic curves given by a real model use $D_\infty = g\infty^+$ instead of the divisor D_∞ we used in the previous section [79, 80]. In particular, this implies that the points ∞^+ and ∞^- need to be k-rational.

A simple modification of Algorithm 3.7 can be used to add divisors in $\mathrm{Cl}^0(C)$ using $D_\infty = g\infty^+$ as base divisor. All one needs to do is change the finishing condition in step 4 from $((\omega^+ < g/2)$ or $(\omega^- < (g-1)/2))$ to $(\omega^+ > g)$. Following the analysis made in the previous section, we know that

$$\tilde{D}_1 + \tilde{D}_2 \equiv D + \omega^+\infty^+ + \omega^-\infty^-,$$

which using $D_\infty = g\infty^+$ becomes

$$\tilde{D}_1 - g\infty^+ + \tilde{D}_2 - g\infty^+ \equiv (D + (\omega^+ - g)\infty^+ + \omega^-\infty^-) - g\infty^+.$$

The condition $(\omega^+ > g)$ guarantees that the divisor $E = D + (\omega^+ - g)\infty^+ + \omega^-\infty^-$ is effective, and the algorithm returns the canonical representative of the corresponding divisor class.

Indeed, one can verify that using Algorithm 3.7 with the modified terminating condition coincides with the addition algorithms presented in [79, 80].

We will now compare the two proposals for addition algorithms on $\mathrm{Cl}^0(C)$. Since the performance of the algorithms, specially for cryptographic applications, will depend exclusively on its behaviour when adding generic divisors, we will restrict our analysis to this case.

Even genus

Assume for a moment that the curve C has even genus g, and that D_1 and D_2 are two effective affine divisors of degree g. Generically, the result D_3 of applying successive reductions to $\mathrm{comp}(D_1, D_2)$ until the degree is at most $g+1$ is a divisor D_3 of degree g. If this is the case, we have

$$D_1 + D_2 \equiv D_3 + (g/2)(\infty^+ + \infty^-), \tag{3.3.1}$$

Notice that the counterweights between $D_1 + D_2$ and D_3 are equal, this is a consequence of Equation (3.2.1). Using Equation (3.3.1) with $D_\infty = (g/2)(\infty^+ + \infty^-)$, we get

$$D_1 - D_\infty + D_2 - D_\infty \equiv D_3 - D_\infty,$$

which means that we have found the result of adding $D_1 - D_\infty$ and $D_2 - D_\infty$, and no "composition at infinity and reduction" steps were necessary.

If instead we work with a divisor at infinity $D'_\infty = g\infty^+$, Equation (3.3.1) becomes

$$D_1 - D'_\infty + D_2 - D'_\infty = D_3 - D'_\infty - (g/2)(\infty^+ - \infty^-),$$

so typically one will need $g/2$ extra red_∞ steps to find D_4 such that

$$D_4 - D'_\infty = (D_1 - D'_\infty) + (D_2 - D'_\infty),$$

it is not difficult to see that the need for the red_∞ steps is related to the fact that the valuations of D'_∞ at the two points at infinity are so different.

Example 3.3.2. We will continue using the data from Example 3.3.1. We were given two divisors D_1 and D_2, and we found a divisor D_4 such that

$$[D_1 - (\infty^+ + \infty^-)] + [D_2 - (\infty^+ + \infty^-)] = [D_4 - (\infty^+ + \infty^-)].$$

To add divisor classes using $2\infty^+$ as base divisor, we need to perform an extra red_∞ step on D_4.

$$\mathrm{red}_\infty(D_4) = D_5, (1, -1)$$
$$D_5 = \mathrm{div}(x^2 + 48x + 34, 17x + 89).$$

We know that

$$D_1 + D_2 \equiv D_5 + 2\infty^+,$$

which implies

$$[D_1 - 2\infty^+] + [D_2 - 2\infty^+] = [D_5 - 2\infty^+].$$

The situation presented in this example, where extra composition-and-reduction steps are needed to add divisors using an unbalanced base divisor is typical, and represents the expected behaviour of the addition algorithms.

Odd genus

Now consider a curve C of odd genus g, and let again D_1 and D_2 be degree g affine divisors. Typically, the result after step 2 in Algorithm 3.7 on the divisors D_1 and D_2 will be a divisor D_3 of degree $g + 1$ such that

$$D_1 + D_2 \equiv D_3 + \frac{g-1}{2}(\infty^+ + \infty^-). \tag{3.3.2}$$

Again, the counterweights between $D_1 + D_2$ and D_3 are equal as a consequence of Equation (3.2.1), and if we now compute $D_4 = \mathrm{red}_\infty(D_3)$, then generically

$$D_3 \equiv D_4 + \infty^-,$$

which together with Equation (3.3.2) gives us

$$D_1 + D_2 \equiv D_4 + \frac{g+1}{2}\infty^+ + \frac{g-1}{2}\infty^-. \tag{3.3.3}$$

Using our base divisor $D_\infty = (g+1)/2\infty^+ + (g-1)/2\infty^-$, we get

$$[D_1 - D_\infty] + [D_2 - D_\infty] = [D_4 - D_\infty],$$

and only one red_∞ step was needed. Notice that in this case the addition algorithm consists of composition, a series of standard reduction steps, and the last step is a single application of red_∞.

Using the base divisor $D'_\infty = g\infty^+$, Equation (3.3.2) becomes

$$D_1 - D'_\infty + D_2 - D'_\infty \equiv D_3 - D'_\infty - (g-1)/2(\infty^+ - \infty^-),$$

so one will typically need $(g-1)/2$ extra steps to find D_4 such that

$$[D_4 - D'_\infty] = [D_1 - D'_\infty] + [D_2 - D'_\infty].$$

Again, the need for the red_∞ steps stems from the difference in the valuations of D_∞ at both points at infinity.

Example 3.3.3. Let C be the hyperelliptic curve defined over $\mathbf{F}_{211}$ by the

equation $C : y^2 = x^8 + 53x^5 + 158x^4 + 12x^3 + x + 187$. Let

$$D_1 = \mathrm{div}(x^3 + 40x^2 + 28x + 134, x^4 + 91x^2 + 143x + 92)$$
$$D_2 = \mathrm{div}(x^3 + 110x^2 + 104x + 197, x^4 + 93x^2 + 52x + 50)$$

be two divisors on C. We have

$$\begin{aligned}\mathrm{comp}(D_1, D_2) &= D_3, (0, 0)\\ D_3 &= \mathrm{div}(x^6 + 150x^5 + 101x^4 + 186x^3 + x^2 + 40x + 23,\\ &\qquad 47x^5 + 169x^4 + 155x^3 + 209x^2 + 161x + 166).\end{aligned}$$

$$\begin{aligned}\mathrm{red}(D_3) &= D_4, (1, 1)\\ D_4 &= \mathrm{div}(x^4 + 149x^3 + 129x^2 + 152x + 198,\\ &\qquad 20x^3 + 155x^2 + 57x + 56)\\ \mathrm{red}_\infty(D_4) &= D_5, (1, 0)\\ D_5 &= \mathrm{div}(x^3 + 195x^2 + 181x + 5, 102x^2 + 154x + 38)\\ \mathrm{red}_\infty(D_5) &= D_6, (1, -1)\\ D_6 &= \mathrm{div}(x^3 + 93x^2 + 138x + 147, 113x^2 + 83x + 137).\end{aligned}$$

This implies that

$$D_1 + D_2 \equiv D_5 + 2\infty^+ + \infty^-$$
$$D_1 + D_2 \equiv D_6 + 3\infty^+$$

If $D_\infty = (2\infty^+ + \infty^-)$ and $D'_\infty = 3\infty^+$, then

$$[D_1 - D_\infty] + [D_2 - D_\infty] = [D_5 - D_\infty]$$
$$[D_1 - D'_\infty] + [D_2 - D'_\infty] = [D_6 - D'_\infty].$$

Which shows that adding divisors using a non-balanced base divisor is more expensive.

Remark 3.3.3. If the curve C has odd genus g, one could try to recover a balanced representation of its elements by representing them as $D_0 - (g + 1)/2(\infty^+ + \infty^-)$, where D_0 is a degree $g + 1$ effective divisor with reduced affine part. This representation could be useful in the intermediate steps of a given implementation, since the use of the red_∞ algorithm would be limited to the final step of the calculation. When divisors are represented using this approach, the number of coefficients needed to describe them increases, and the composition formulae get more complicated.

We have seen that using a "balanced" divisor at infinity, generically the number of red_∞ steps needed to compute the addition of two divisor classes in $\mathrm{Cl}^0(C)$ is 0 when g is even and 1 when g is odd; whereas when using a non-balanced divisor, the number of red_∞ steps needed to compute the addition of two divisors is generically $g/2$ for even g and $(g-1)/2$ for odd g.

In order to compare the two proposals for arithmetic in $\mathrm{Cl}^0(C)$, we must also consider the computation of inverses, a fundamental operation in a computable group which has, surprisingly, been ignored in the literature. Besides its trivial use to invert divisors, this operation is fundamental to achieve fast divisor multiplication through signed representations.

Divisor inversion

We will just analyse inversion in the generic case. To do this let D be a degree g affine effective divisor on C. Assume for a moment that g is even. The inverse of the divisor $P = D - (g/2)(\infty^+ + \infty^-)$ is the divisor $\overline{D} - (g/2)(\infty^+ + \infty^-)$, whereas if we now assume that g is odd, the divisor

$$(D - \frac{g+1}{2}\infty^+ - \frac{g-1}{2}\infty^-) + (\overline{D} - \frac{g-1}{2}\infty^+ - \frac{g+1}{2}\infty^-)$$

is principal, which means that $\overline{D} - (g-1)/2\infty^+ - (g+1)/2\infty^-$ is the inverse of P, and in order to fix the divisor at infinity, using Proposition 3.2.1 it is easy to see that generically only one application of Algorithm 3.6 will suffice. In other words, using the "balanced" representation at infinity, 0 or 1 applications of Algorithm 3.6 will be needed, depending on the parity of g.

We now analyse the computation of inverses using $D'_\infty = g\infty^+$ as base divisor. Clearly, the divisor

$$(D - g\infty^+) + (\overline{D} - g\infty^-)$$

is principal, so we need to find an appropriate representative of the divisor class $[\overline{D} - g\infty^-]$. Again, this can be done through g applications of Algorithm 3.6, as can be easily seen using Proposition 3.2.1.

It is now clear that computing the inverse of a divisor class is easier when the divisor at infinity is as balanced as possible, supporting our claim that a "balanced" representation is a closer analogue to that of Cantor for imaginary models, where the inverse of a divisor is its hyperelliptic conjugate, just as in our case when the genus of C is even.

Table 3.1 gives the cost of addition and doubling in a genus 2 curve using

	Imaginary	Balanced	Non-balanced
Addition	1I, 2S, 22M [63]	1I, 2S, 26M	2I, 4S, 30M
Doubling	1I, 5S, 22M [63]	1I, 4S, 28M	2I, 6S, 32M
Inversion	0	0	2I, 4S, 8M

Table 3.1: Operation counts for genus 2 arithmetic using formulae of [26] .

the explicit formulae for Algorithms 3.4, 3.5 and 3.6 presented in [26]. If $S = M$ and $I = 4M$ then balanced representations give a saving of around 15% for addition and 13% for doubling (if $I = 30M$ the savings become 62% and 58% respectively). The extra operations in the non-balanced case come from an additional application of Algorithm 3.6 in each case.

3.4 Appendix: Explicit 'imaginary baby steps'

In Remark 3.2.4 we mentioned that the analogue of a baby step for curves given by an imaginary model would be the addition of a point of the form $P - \infty$, where P has a special form. In Algorithm 3.9, we present an optimized algorithm to compute these "imaginary baby steps" in a genus 2 curve C defined over the field k by

$$C : y^2 = x^5 + \sum_{i=0}^{4} f_i x^i,$$

where we further assume that f_0 is a square in k, so that the point $P = (0, y_b)$ on C is k-rational. Divisors will be represented by their Mumford representation $(x^2 + u_1 x + u_0, v_1 x + v_0)$.

The cost of a "real baby step" in genus 2 is (1I,2S,4M) [26] and Algorithm 3.9 needs (1I,1S,5M), it is therefore a very efficient analogue of Algorithm 3.6 for curves given by an imaginary model.

Algorithm 3.9 Imaginary Baby Step

INPUT: $D = \text{div}(x^2 + u_1 x + u_0, v_1 x + v_0)$.
OUTPUT: $D_2 = \text{div}(u_2, v_2)$ such that $[D_2] = [D] + [(0, y_b) - \infty]$.
1: $\mu := (y_b - v_0)/u_0$ (1I,1M).
2: $b1 := f_4 - \mu^2 - u_1$ (1S).
3: $b_0 := f_3 - u_0 - 2\mu v_1 + u_1(u_1 - \mu^2 - f_4)$ (2M).
4: $c_1 := v_1 + \mu(u_1 - b_1)$ (1M).
5: $c_0 := v_0 + \mu(u_0 - b_0)$ (1M).
6: **return** $\text{div}(x^2 + b_1 x + b_0, -c_1 x - c_0)$.

We would like to point out that the idea of using degenerate divisors (a divisor is degenerate if it has less than g affine points in its support) has been considered before. Katagi et.al. [58, 59] analysed the advantages of using degenerate divisors in cryptographyc applications, and the use of degenerate divisors to optimise pairing computations has also been discussed in [3, 31, 38]. However, the use of a point P with a special form in the degenerate divisor seems to have been overlooked.

Chapter 4

Infrastructure

In this chapter we will recall the definition of the set of infrastructure ideals in the function field of a real hyperelliptic curve, we present the applications of infrastructure to cryptography, and we conclude by relating the set of infrastructure ideals with the class group of the corresponding hyperelliptic curve.

Our main result is Theorem 4.5.3, proving that there exists a map from the set of infrastructure ideals into the class group of the underlying hyperelliptic curve that preserves the "group-like"structure of the infrastructure. As a consequence of this result we show that calculating distances in the set of infrastructure ideals is equivalent to the DLP in the underlying hyperelliptic curve. This is a significant contribution as this result was previously known only in genus 1. The results of this chapter have been presented as [72].

4.1 Historical overview

Let K be a quadratic number field and let $\mathcal{O}$ be an order in K with discriminant D. If K is an imaginary number field, there is a well-known bijection between the set of *reduced, positive definite* quadratic forms $\mathbf{F}$, and the ideal

class group $\mathrm{Cl}(\mathcal{O})$ of $\mathcal{O}$ (see Lenstra [66]).

When K is a real quadratic field this bijection no longer exists. Instead, there is a many-to-one map $\psi : \mathbf{F} \longrightarrow \mathrm{Cl}(\mathcal{O})$ from the group of reduced quadratic forms $\mathbf{F}$ to the ideal class group $\mathrm{Cl}(\mathcal{O})$. Shanks realized that the set $\mathcal{R} = \psi^{-1}(\mathcal{O})$ of quadratic forms mapping into the principal class could be given an algebraic structure, associating a *distance* to every element [90].

The set $\mathcal{R}$, together with the underlying structure described by Shanks is known as the infrastructure of $\mathcal{O}$. The fastest algorithms to compute the regulator of $\mathcal{O}$ known to-date use infrastructure.

Buchmann and Williams presented a key exchange algorithm very similar to the Diffie-Hellman proposal [22] using the infrastructure of an order $\mathcal{O}$ in a real number field K [12]. This proposal had a number of problems, including the need to deal with approximations to certain algebraic numbers, high bandwidth and ambiguity problems. These problems were overcome with the proposal by Scheidler, Stein and Williams in [86] to use the set of infrastructure ideals in the function field associated to a hyperelliptic curve given by a real model over a finite field. It is this proposal, and its successive adaptations[85, 56, 55], that we study in this chapter.

4.2 Infrastructure

Let C be a genus g hyperelliptic curve given by a real model $C : y^2 = F(x)$ over a field k with $\mathrm{char}(k) \neq 2$. Denote its function field as $K = k(C)$, and let $\mathcal{O}$ be the affine coordinate ring of C, i.e. $\mathcal{O} = k[x, y]/(y^2 - F(x))$.

Every ideal $\mathfrak{a}$ of $\mathcal{O}$ has an $\mathcal{O}$-basis of the form $\mathfrak{a} = [SQ, S(y + P)]$, where S, Q, P are polynomials in $k[x]$ such that Q divides $P^2 - F$. The polynomials

S and Q are uniquely defined up to multiplication by elements of k^*, and the polynomial P is only defined modulo Q. To have a unique basis for the ideal $\mathfrak{a}$ we will assume that $\deg(P) < \deg(Q)$.

Remark 4.2.1. Throughout this chapter, when we refer to the basis of an ideal we will assume that the basis has this form.

Definition 4.2.2. *If the ideal $\mathfrak{a}$ has basis $[SQ, S(y+P)]$ as described in the previous paragraph, we define the* degree *of the ideal $\mathfrak{a}$ as* $\deg(SQ)$.

Definition 4.2.3. *We say that an ideal is* primitive *if the polynomial S can be taken to be $S = 1$.*

Definition 4.2.4. *We say that an ideal $\mathfrak{a} = [Q, P+y]$ is* reduced *if it is primitive and $g \geq \deg(Q)$.*

Definition 4.2.5. *Let $\mathcal{R}$ be the set of principal reduced ideals of $\mathcal{O}$. We say that $\mathcal{R}$ is the set of infrastructure ideals of $\mathcal{O}$ [86].*

Definition 4.2.6. *Let $\mathfrak{a}$ be an infrastructure ideal. By definition $\mathfrak{a} = (\alpha)$ for some function α. We define the distance $\delta(\mathfrak{a})$ of the ideal $\mathfrak{a}$ as $\delta(\mathfrak{a}) = \mathrm{ord}_{\infty^+}(\alpha)$ [86].*

Example 4.2.1. The ring $\mathcal{O}$ can be seen as the ideal generated by the element 1. It is an element of $\mathcal{R}$ with basis $\mathcal{O} = [1, 0]$, and by definition it has distance 0.

If there is a unit β in $\mathcal{O}$ with non-zero valuation at ∞^+, then there is a least positive integer R for which there exists a unit β_R with valuation R at ∞^+. In this case, the distance of an ideal is only defined modulo R. The integer R is known as the *regulator* of $\mathcal{O}$.

Remark 4.2.7. The divisor of a unit β in $\mathcal{O}$ has to be supported exclusively at ∞^+ and ∞^-, and have degree 0. It follows that the regulator of $\mathcal{O}$ is given by the order of the element $\infty^+ - \infty^-$ in $\mathrm{Cl}^0(C)$. We prove some stronger results in Theorems 4.5.3 and 4.5.5.

Ideas related to the set of infrastructure ideals have found their main applications in cryptography. For these applications the curve C is defined over a finite field and the set $\mathcal{R}$ is finite.

Definition 4.2.8. *Given ideals $\mathfrak{a}_1, \mathfrak{a}_2 \in \mathcal{R}$, we define the* distance between $\mathfrak{a}_1$ and $\mathfrak{a}_2$ *to be*

$$\delta(\mathfrak{a}_1, \mathfrak{a}_2) = \delta(\mathfrak{a}_1) - \delta(\mathfrak{a}_2).$$

Given two random infrastructure ideals $\mathfrak{a}_1$ and $\mathfrak{a}_2$, finding the distance between them is a hard problem (see Theorem 4.5.8). A very good description of the ideas and techniques used in the infrastructure of a hyperelliptic curve given by a real model can be found in [79].

4.3 Operations on the Infrastructure Ideals

Let $\mathfrak{a}$ be a primitive ideal of $\mathcal{O}$ with basis $[Q, y + P]$. Note that the pair of polynomials (Q, P) satisfy all the conditions to be the Mumford representation of a divisor. In other words, there is an effective, affine, semi-reduced divisor D on the curve C such that $D = \mathrm{div}(Q, P)$.

Definition 4.3.1. *Given a primitive ideal $\mathfrak{a}$ of $\mathcal{O}$ with basis $[Q, y + P]$, we define the divisor associated to $\mathfrak{a}$ as the divisor $D = \mathrm{div}(\mathfrak{a})$ whose Mumford representation is (Q, P).*

Since the basis $[Q, y + P]$ of the ideal $\mathfrak{a}$ could also be thought of as the

Mumford representation of a divisor, we can use Algorithm 3.4 (composition), Algorithm 3.5 (reduction) and Algorithm 3.6 (composition at infinity and reduction) on elements of $\mathcal{R}$. The idea of using these algorithms (or rather, a variant that does not compute counterweights) on ideals is not new (see [13, 79]). The interpretation of the action of these algorithms on infrastructure ideals is not obvious. In this section we give an interpretation both of the action of the algorithms on ideals of $\mathcal{O}$, and of the counterweights returned by the algorithms in terms of the distance.

Proposition 4.3.2. *Let $\mathfrak{a}_1$ and $\mathfrak{a}_2$ be two primitive ideals of $\mathcal{O}$. If $\mathfrak{a}_3, (\omega^+, \omega^-) = \mathrm{comp}(\mathfrak{a}_1, \mathfrak{a}_2)$ is the result obtained from applying Algorithm 3.4 on the basis of $\mathfrak{a}_1$ and $\mathfrak{a}_2$, we get*

$$\mathfrak{a}_1 \cdot \mathfrak{a}_2 = s \cdot \mathfrak{a}_3$$
$$\omega^+ = \mathrm{ord}_{\infty^+}(s),$$

where s denotes the polynomial obtained in Step 1 of Algorithm 3.4.

Proof. The first property is a classic result. See [86, Theorem 3.4]. The second follows from the fact that the order of a polynomial $p(x)$ at ∞^+ is given by the degree $\deg p(x)$.

□

Proposition 4.3.3. *Let $\mathfrak{a}_0$ be a primitive divisor with basis $\mathfrak{a}_0 = [Q_0, y + P_0]$ such that $\deg(Q_0) \geq g + 2$. If we let $\mathfrak{a}_1, (\omega^+, \omega^-) = \mathrm{red}(\mathfrak{a}_0)$ be the result of*

applying Algorithm 3.5 to the ideal $\mathfrak{a}_0$*, then*

$$\begin{aligned}\mathfrak{a}_1 &= \left(\frac{y - P_0}{Q_0}\right) \cdot \mathfrak{a}_0 \\ \omega^+ &= -\operatorname{ord}_{\infty^+}\left(\frac{y - P_0}{Q_0}\right).\end{aligned}$$

Proof. If the ideal $\mathfrak{a}_0$ has basis $[Q_0, y + P_0]$, then the ideal $((y - P_0)/Q_0) \cdot \mathfrak{a}_0$ has basis $[y - P_0, (y^2 - P_0^2)/Q_0]$, which is by definition a basis of $\mathfrak{a}_1$. The second assertion follows simply from Equation (3.2.3). □

Proposition 4.3.4. *Let* $\mathfrak{a}_0$ *be a primitive divisor with basis* $\mathfrak{a}_0 = [Q_0, y + P_0]$ *such that* $\deg(Q_0) \leq g + 1$*. If we let* $\mathfrak{a}_1, (\omega^+, \omega^-) = \operatorname{red}_\infty(\mathfrak{a}_0)$ *be the result of applying Algorithm 3.6 to the ideal* $\mathfrak{a}_0$*, then*

$$\begin{aligned}\mathfrak{a}_1 &= \left(\frac{y - P_0}{Q_0}\right) \cdot \mathfrak{a}_0, \\ \omega^+ &= -\operatorname{ord}_{\infty^+}\left(\frac{y - P_0}{Q_0}\right).\end{aligned}$$

Proof. The proof of the first property is analogous to the proof given for Proposition 4.3.3. The second property follows from Equation (3.2.5). □

Corollary 4.3.5. *Let* $\mathfrak{a}_0$ *be an infrastructure ideal. If* $\mathfrak{a}_1, (\omega^+, \omega^-) = \operatorname{red}_\infty(\mathfrak{a}_0)$*, then*

$$\omega^+ = -\delta(\mathfrak{a}_0, \mathfrak{a}_1).$$

Proof. Proposition 4.3.4 shows that there is a function α with $\omega^+ = -\operatorname{ord}_{\infty^+}(\alpha)$ such that $\mathfrak{a}_1 = \alpha\mathfrak{a}_0$, the result follows from the definition of δ. □

Definition 4.3.6. *Suppose that the regulator* R *is a positive integer. Given*

an integer n between 0 and $R-1$, let

$$\delta(n) = \max\{\delta(\mathfrak{a}) | \mathfrak{a} \in \mathcal{R} \text{ and } \delta(\mathfrak{a}) \leq n\},$$

and let $\mathfrak{a}_n$ be the ideal in $\mathcal{R}$ such that $\delta(\mathfrak{a}_n) = \delta(n)$. We say that $\mathfrak{a}_n$ is the ideal closest to the left *of n [86].*

The following result shows that in principle it is possible to find all the infrastructure ideals using only the algorithms we have presented, we omit the proof, but refer the reader to [86].

Proposition 4.3.7. *Let $\mathfrak{a}$ be an infrastructure ideal. Then the set $\{\mathrm{red}_\infty^i(\mathfrak{a})\}_{i \in \mathbf{Z}}$ is the set of infrastructure ideals $\mathcal{R}$.*

Proof. See [86, Section 3.1]. □

4.4 A Cryptographic Interlude

The cryptographic applications of infrastructure have been the motivation for most of the work done in the area. In this section we present the cryptographic protocols presented in [86] which use the set of infrastructure ideals as underlying algebraic structure. It has been claimed that this is the unique Diffie-Hellman-like key exchange protocol that doesn't use a group as underlying algebraic structure, we analyse this claim in the next section, see for example Theorem 4.5.3.

Given an infrastructure ideal $\mathfrak{a}_0$ with distance δ_0 and an integer k, Algorithm 4.1 finds the ideal closest to the left of $k + \delta_0$. We denote the result of Algorithm 4.1 as $\mathfrak{a}_1 = \mathrm{CA}(\mathfrak{a}_0, k)$.

Algorithm 4.1 Constant Addition

INPUT: An ideal $\mathfrak{a}_0 \in \mathcal{R}$ and an integer k.
OUTPUT: The ideal $\mathfrak{a}_1$ closest to the left of $\delta(\mathfrak{a}_0) + k$.

1: **if** k is positive **then**
2: Use H^+ in the red_∞ steps.
3: Let $\mathfrak{a}_2, (\omega^+, \omega^-) := \text{red}_\infty(\mathfrak{a}_0)$.
4: Let $\mathfrak{a}_1 := \mathfrak{a}_0,$, $n := \omega^+$.
5: **while** $n < k$ **do**
6: $\mathfrak{a}_1 := \mathfrak{a}_2$.
7: $\mathfrak{a}_2, (\omega^+, \omega^-) := \text{red}_\infty(\mathfrak{a}_0)$, $n := n + \omega^+$.
8: **else if** k is negative **then**
9: Use H^- in the red_∞ steps.
10: Let $\mathfrak{a}_2, (\omega^+, \omega^-) := \text{red}_\infty(\mathfrak{a}_0)$.
11: Let $\mathfrak{a}_1 := \mathfrak{a}_0$, $n := \omega^+$.
12: **while** $n \geq k$ **do**
13: $\mathfrak{a}_1 := \mathfrak{a}_2$.
14: $\mathfrak{a}_2, (\omega^+, \omega^-) := \text{red}_\infty(\mathfrak{a}_0)$, $n := n + \omega^+$.
15: **return** $\mathfrak{a}_1$.

It can be proved that the maximum value of n in Step 8 in Algorithm 4.2 is bounded by $(g+1)/2$ (see [86]). Algorithm 4.2 shows that given two ideals with distances δ_1 and δ_2, it is possible to find the ideal closest to the left of $\delta_1 + \delta_2$.

Algorithm 4.2 Ideal Multiplication

INPUT: Ideals $\mathfrak{a}_1, \mathfrak{a}_2 \in \mathcal{R}$.
OUTPUT: The ideal $\mathfrak{a}_3$ closest to the left of $\delta(\mathfrak{a}_1) + \delta(\mathfrak{a}_2)$.

1: $\mathfrak{a}_3, (\omega^+, \omega^-) := \text{comp}(\mathfrak{a}_1, \mathfrak{a}_2)$, $n := \omega^+$.
2: **while** $\deg(\mathfrak{a}_3) \geq g + 2$ **do**
3: $\mathfrak{a}_3, (\omega^+, \omega^-) := \text{red}(\mathfrak{a}_3)$, $n := n + \omega^+$.
4: **if** $\deg(\mathfrak{a}_3) = g + 1$ **then**
5: $\mathfrak{a}_3, (\omega^+, \omega^-) := \text{red}_\infty(\mathfrak{a}_3)$, $n := n + \omega^+$.
6: $\mathfrak{a}_3 := \text{CA}(\mathfrak{a}_3, n)$.
7: **return** $\mathfrak{a}_3$.

Combining Algorithm 4.2 with Algorithm 4.1, given an infrastructure ideal $\mathfrak{a}$ with distance δ and an integer k, it is possible to find the ideal closest to the left of $k\delta$, even if δ is not known, in time $O(\log(k))$ (see Algorithm POWER

in [86]). This construction can then be used to implement a key-exchange protocol modeled on Diffie-Hellman using the infrastructure ideals.

Algorithm 4.3 Infrastructure Diffie-Hellman

PUBLIC INFORMATION: An ideal $\mathfrak{a} \in \mathcal{R}$ and its distance $\delta = \delta(\mathfrak{a})$.

1: Alice generates a random ideal $\mathfrak{a}_A$ with distance $\delta(\mathfrak{a}_A) = a\delta$. Alice knows a.
2: Bob generates a random ideal $\mathfrak{a}_B$ with distance $\delta(\mathfrak{a}_B) = b\delta$. Bob knows b.
3: Alice and Bob exchange $\mathfrak{a}_A$ and $\mathfrak{a}_B$.
4: Alice and Bob compute the ideal $\mathfrak{a}_C$ closes on the left to $ab\delta$.
5: Alice and Bob use $\mathfrak{a}_C$ as the key in a symmetric encryption scheme.

In the context of affine semi-reduced effective divisors, we mentioned that Algorithm 3.6 (composition at infinity and reduction) has the same action on a divisor D as Algorithm 3.5 (reduction) if $\deg(D) \geq g + 2$. In the context of infrastructure, it is customary to use only Algorithm 3.6; we have chosen to differentiate its use in Algorithm 4.2 precisely to separate the cases when only a reduction is taking place from those when some composition at infinity is also carried out. From a practical perspective the algorithms are identical, but we believe that conceptually they deserve being treated differently.

4.5 A map into the class group

As mentioned before, it is claimed that Algorithm 4.3 provides a Diffie-Hellman-type key construction algorithm in a non-group structure. In this section we will explain that the failure of $\mathcal{R}$ to be a group is somewhat artificial. The results in this section are based on the construction of a map relating the set of infrastructure ideals with certain divisor classes in $\mathrm{Cl}^0(C)$.

Definition 4.5.1. *Given $\mathfrak{a} \in \mathcal{R}$ an infrastructure ideal, define*

$$\begin{aligned} \psi : &\mathcal{R} \longrightarrow \mathrm{Cl}^0(C) \\ &\mathfrak{a} \mapsto [\mathrm{div}(\mathfrak{a}) - \deg(\mathfrak{a})\infty^-], \end{aligned}$$

where $\mathrm{div}(\mathfrak{a})$ *refers to the affine effective semi-reduced divisor associated to* $\mathfrak{a}$ *(see Definition 4.3.1).*

Proposition 4.5.2. *Let $\mathfrak{a}_1$ and $\mathfrak{a}_2$ be two infrastructure ideals. If δ is the distance $\delta(\mathfrak{a}_1, \mathfrak{a}_2)$ between $\mathfrak{a}_1$ and $\mathfrak{a}_2$, then*

$$\psi(\mathfrak{a}_1) + \delta[\infty^+ - \infty^-] = \psi(\mathfrak{a}_2). \tag{4.5.1}$$

Proof. Proposition 4.3.7 shows that one can reach any element of $\mathcal{R}$ using successive applications of red_∞ on $\mathfrak{a}_1$, so it suffices to prove this result for ideals $\mathfrak{a}_1$ and $\mathfrak{a}_2, (\omega^+, \omega^-) = \mathrm{red}_\infty(\mathfrak{a}_1)$. Let $D_1 = \mathrm{div}(\mathfrak{a}_1)$ and $D_2 = \mathrm{div}(\mathfrak{a}_2)$ be affine divisors of degrees d_1 and d_2 respectively. Step 4 in Algorithm 3.6 says that $(\omega^+, \omega^-) = (d_1 - g - 1, g + 1 - d_2)$, and using Equation (3.2.5) we get

$$D_1 \equiv D_2 + (d_1 - (g+1))\infty^+ + (g + 1 - d_2)\infty^-.$$

This implies

$$(D_1 - d_1\infty^-) + (g + 1 - d_1)(\infty^+ - \infty^-) \equiv (D_2 - d_2\infty^-),$$

Since $\mathfrak{a}_1 = \mathrm{red}_\infty(\mathfrak{a}_0)$, Corollary 4.3.5 proves that $\delta = -\omega^+$, and since $\omega^+ =$

$d_1 - g - 1$. We can finally conclude that

$$\psi(a) + \delta(\infty^+ - \infty^-) = \psi(b).$$

□

Theorem 4.5.3. *The map $\psi : \mathcal{R} \longrightarrow \mathrm{Cl}^0(C)$ sends an ideal $\mathfrak{a}$ with distance $\delta = \delta(\mathfrak{a})$ to the element $\psi(\mathfrak{a}) = \delta[\infty^+ - \infty^-]$ of $\mathrm{Cl}^0(C)$.*

Proof. The result is trivial for $\mathfrak{a} = \mathcal{O}$, since we have mentioned that $\deg(\mathcal{O}) = 0$ and by definition $\psi(\mathcal{O}) = 0$. It extends to $\mathcal{R}$ using Proposition 4.5.2. □

Corollary 4.5.4. *Let $\mathfrak{a}$ be an infrastructure ideal with distance δ, then the Mumford representation of the affine part of the canonical representative of $\delta[\infty^+ - \infty^-]$ using base divisor $D_\infty = g\infty^-$ is given by the polynomials in the basis of $\mathfrak{a}$.*

We have proved that there is a simple map ψ sending the infrastructure ideals into the class group $\mathrm{Cl}^0(C)$ that is compatible with the *group-like* structure of $\mathcal{R}$. Using the explicit description of ψ, we can describe exactly the elements missing in $\mathcal{R}$ to be a group. Let $\mathrm{div}((u, v), n)$ denote (see Definition 3.3.1) the element

$$\mathrm{div}(u, v) + n\infty^+ + (g - \deg(u) - n)\infty^- - g\infty^-,$$

and let $\mathbf{G} = \langle[\infty^+ - \infty^-]\rangle$ be the group generated by $[\infty^+ - \infty^-]$. Using this notation we have the following

Theorem 4.5.5. *The image $\psi(\mathcal{R})$ of the infrastructure ideals under ψ, consists of the elements of $\mathbf{G}$ of the form $\mathrm{div}((u, v), 0)$.*

Proof. A different way to state this theorem is by saying that a divisor class $[D]$ in $\mathbf{G}$ is in the image of ψ if and only if the coefficient of ∞^+ in its canonical representative with base divisor $g\infty^-$ is zero.

By construction, all ideals in the image $\psi(\mathcal{R})$ have the indicated form. We will prove the converse by induction, and we will only prove it for positive multiples of $[\infty^+ - \infty^-]$, as the proof for negative multiples is either not necessary or analogous.

Let $\mathfrak{a}_0$ be an ideal with distance $\delta_0 = \delta(\mathfrak{a}_0)$. Denote $\mathfrak{a}_1, (\omega^+, \omega^-) = \mathrm{red}_\infty(\mathfrak{a}_0)$ and let $\delta_1 = \delta(\mathfrak{a}_1)$. Since $\omega^+ = \deg(\mathfrak{a}_0) - g - 1$, Corollary 4.3.5 proves that $\delta_1 - \delta_0 = g + 1 - \deg(\mathfrak{a}_0)$. We will show that none of the elements $n[\infty^+ - \infty^-]$, for $\delta_0 < n < \delta_1$, has the indicated form (if $\delta_1 - \delta_0 = 1$ this is a vacuous statement).

We know from Theorem 4.5.3 that $\psi(\mathfrak{a}_0) = \delta_0[\infty^+ - \infty^-]$, and by definition

$$\psi(\mathfrak{a}_0) = [\mathrm{div}(\mathfrak{a}_0) + (g - \deg(\mathfrak{a}_0))\infty^- - g\infty^-].$$

For every n such that $\delta_0 < n < \delta_1$ the divisor

$$\big(\mathrm{div}(\mathfrak{a}_0) + (g - \deg(\mathfrak{a}_0))\infty^- - g\infty^-\big) + (n - \delta_0)(\infty^+ - \infty^-),$$

gives a representative of the divisor class $n[\infty^+ - \infty^-]$. This divisor can be rewritten as

$$\mathrm{div}(\mathfrak{a}_0) + (n - \delta_0)\infty^+ + (g - \deg(\mathfrak{a}_0) - n + \delta_0)\infty^- - g\infty^-. \tag{4.5.2}$$

Since $\delta_0 < n < \delta_1$ and $\delta_1 - \delta_0 = g + 1 - \deg(\mathfrak{a}_0)$, we have $n - \delta_0 > 0$, and $\deg(\mathfrak{a}_0) - g - n + \delta_0 \geq 0$. It follows that the divisor given by Equation (4.5.2) is

the canonical representative of $n[\infty^+ - \infty^-]$ in $\mathrm{Cl}^0(C)$ with base divisor $g\infty^-$. But the coefficient of ∞^+ in this divisor is not zero, hence it does not have the form $\mathrm{div}((u,v),0)$ and the result follows. □

Corollary 4.5.6. *Let $\mathfrak{a}_1, \mathfrak{a}_2$ and $\mathfrak{a}_3$ be infrastructure ideals with*

$$\delta(\mathfrak{a}_1) + \delta(\mathfrak{a}_2) = \delta(\mathfrak{a}_3),$$

then the operations needed to calculate $\mathfrak{a}_3$ from $\mathfrak{a}_1$ and $\mathfrak{a}_2$ are the same as the operations needed to add $\delta(\mathfrak{a}_1)[\infty^+ - \infty^-]$ and $\delta(\mathfrak{a}_2)[\infty^+ - \infty^-]$ in $\mathrm{Cl}^0(C)$, when these two ideal classes are given by their canonical representatives with base divisor $D_\infty = g\infty^-$.

Remark 4.5.7. Theorem 4.5.3 and Corollary 4.5.6 show that the use of infrastructure in cryptographic protocols is equivalent to the implementation of these protocols in the class group $\mathrm{Cl}^0(C)$ of the corresponding hyperelliptic curve C, with the disadvantage that the infrastructure has some 'holes', as proven in Theorem 4.5.5, while $\mathrm{Cl}^0(C)$ is a group. Corollary 4.5.6 also shows that the representation of the elements of $\mathrm{Cl}^0(C)$ used when working with the infrastructure is non-optimal, and it would be better to work with the representation using a balanced divisor at infinity as described in the article [36].

It is possible to use our results to properly assess the difficulty of computing the distance of a random infrastructure ideal $\mathfrak{a}$. The only arguments known in this direction show that the problem of finding distances in the set of infrastructure ideals associated to a real model for an elliptic curve E is equivalent to the DLP in E; it is then argued that if an algorithm existed to compute distances in *all* curves, then this algorithm could be used to solve the DLP in an elliptic curve. This argument is not satisfactory, and we now

present a more refined analysis.

Theorem 4.5.8. *Let $\mathcal{R}$ be the set of infrastructure ideals associated to the hyperelliptic curve C given by a real model. The problem of computing the distance $\delta(\mathfrak{a})$ of a random ideal $\mathfrak{a}$ in $\mathcal{R}$ is equivalent to the DLP in the subgroup $\mathbf{G} = \langle [\infty^+ - \infty^-] \rangle$ of the class group $\mathrm{Cl}^0(C)$.*

Proof. Let $\mathfrak{a}$ be an ideal in $\mathcal{R}$ with distance $\delta = \delta(\mathfrak{a})$. Theorem 4.5.3 shows that $\psi(\mathfrak{a}) = \delta[\infty^+ - \infty^-]$. The element $\psi(\mathfrak{a})$ can be computed in polynomial time, and the problem of finding δ is thus reduced to finding the discrete logarithm of $\psi(\mathfrak{a})$ with respect to $[\infty^+ - \infty^-]$.

To prove the reverse implication, note that Theorem 4.5.5 shows that a random element of $\mathbf{G}$ will belong to the image of $\mathcal{R}$ under ψ with high probability. Hence, given two divisor classes $[D_1]$ and $[D_2]$, one can find an integer n relatively prime to the regulator R such that $n[D_1]$ and $n[D_1]$ lie in $\psi(\mathcal{R})$ in probabilistic polynomial time. The map ψ can be inverted in constant time, and if δ_1 and δ_2 are the distances of the ideals $\psi^{-1}(n[D_1])$ and $\psi^{-1}(n[D_1])$, then the discrete logarithm of $[D_2]$ with respect to $[D_1]$ is given by $\delta_2/\delta_1 \mod R$. □

4.6 Conclusions

The main computational applications of infrastructure in the arithmetic of real quadratic number fields are the computation of the regulator and of a fundamental unit. In the case of the infrastructure of a hyperelliptic curve given by a real model over a finite field, there exist efficient algorithms to solve both of these problems.

Calculating the regulator of a hyperelliptic curve C over a finite field $\mathbf{F}_q$

can be done by finding the number of points on the class group $\mathrm{Cl}^0(C)$ of C. The best techniques available to count the number of points on the class group of a hyperelliptic curve do not use the infrastructure of the curve, but rather sofisticated sofisticated algorithms depending on the genus of the curve and the size of the base field [2, 44, 52, 60, 81].

If the regulator of the hyperelliptic curve C is known, the problem of finding a fundamental unit in the affine coordinate ring of a hyperelliptic curve C can be solved in polynomial time using Miller's algorithm. Hence, the only computational task depending on the infrastructure of the curve C would be the Diffie-Hellman-like key exchange algorithm (Algorithm 4.3).

It has been claimed that Algorithm 4.3 provides the unique Diffie-Hellman-like key exchange protocol implemented over a non-group algebraic structure. In this chapter we have shown that there is a simple (and very natural) embedding of the infrastructure ideals into the class group of the curve that makes the group operations in $\mathrm{Cl}^0(C)$ compatible with those of the infrastructure. We have shown that every algorithm using the infrastructure to obtain cryptographic primitives can be implemented more efficiently in the class group of the corresponding hyperelliptic curve C. This is not only because the class group of the curve fills the 'holes' that prevent $\mathcal{R}$ from being a group, but also because the representation of the elements of $\mathrm{Cl}^0(C)$ used when working with the infrastructure is not optimal.

Chapter 5

Pairings on Hyperelliptic Curves with a Real Model

The results presented in this chapter were developed in a joint project with S. Galbraith and X. Lin, and will be published as [39]. In particular, the pairing results reported in Sections 5.3 and 5.4 were implemented by X. Lin, and Algorithm 5.1 was developed by S. Galbraith and X. Lin.

5.1 Introduction

The study of efficient pairing computation on hyperelliptic curves has focused exclusively on the analysis of hyperelliptic curves given by an imaginary model. With the development of new divisor addition algorithms on hyperelliptic curves given by a real model [36], it is natural to ask if pairings can be implemented on these curves competitively.

The authors of [41] construct a genus 2 curve C, defined over $\mathbf{F}_p$ for p a prime $p \equiv 5 \mod 6$. The Jacobian $\mathrm{Jac}(C)$ of this curve has $p^2 - p + 1$ points defined over $\mathbf{F}_p$, and embedding degree 6 with respect to any subgroup with

prime order $r \mid (p^2 - p + 1)$ and $r > 3$. The curve C is given by a real model (see [36]), which in particular means that it has 2 points at infinity.

In [95], Verheul presents the construction of an elliptic curve with embedding degree 3. This curve is defined over a field $\mathbf{F}_{p^2}$ for p a prime $p \equiv 5 \mod 6$, and has $p^2 - p + 1$ points over $\mathbf{F}_{p^2}$. Pairings on these elliptic curves have been studied by Hu et.al. in [51].

The similarities between these curves make them natural candidates for a comparison between elliptic and hyperelliptic curve pairing implementations. In this chapter we explore several optimisation techniques on these curves, implement pairings and compare their performance. Among the optimisations used in the implementation is the recent R-ate pairing proposed by Lee, Lee and Park in [64], and the well-known *denominator elimination* technique, which is combined with the R-ate pairing thanks to Theorem 5.2.4.

A crucial step towards a competitive implementation of pairings on hyperelliptic curves given by a real model is having efficient divisor addition algorithms that result in simple Miller functions. The addition algorithms presented in Chapter 3 allow for a fast implementation not only because the operation count in the addition and doubling algorithms is smaller than that in previous proposals [79], but also because the Miller function, whose evaluation is the bottleneck in pairing computations on high genus curves, is simpler using the algorithms of Chapter 3 than it would be if computed using previous proposals. We make a theoretical and practical comparison of the efficiency of our pairings compared with that of pairings on elliptic and hyperelliptic curves. We conclude that pairings can be efficiently implemented on hyperelliptic curves given by a real model.

This chapter is organized as follows: Section 5.2 presents a specialization of

the results of Chapter 3 to genus 2, and the embedding degree 6 construction of Galbraith, Pujolas, Ritzenthaler and Smith [41]. Section 5.3 describes our parameter generation algorithms and the optimisations used in the implementation. In Section 5.4 we report our implementation results and compare them with pairing computation results obtained for similar elliptic or hyperelliptic curves. Some conclusions are discussed in Section 5.5.

5.2 Genus Two Curves

5.2.1 Arithmetic on hyperelliptic curves

Let C be a genus 2 hyperelliptic curve given by

$$C : y^2 = F(x),$$

where $\mathrm{char}(k) \neq 2, 3$ and $F(x) \in k(x)$ is a square-free degree 6 polynomial. We say that this is a *real model* for C. The desingularization of C has 2 different points at infinity, which we will denote ∞^+ and ∞^- (see Example 2.2.1). Let $D_\infty = \infty^+ + \infty^-$ (see Definition 2.2.10), note that this divisor is k-rational even if the points ∞^+ and ∞^- are not independently so.

Proposition 2.2.9 shows that every element of $\mathrm{Cl}^0(C)$ has a unique representative of the form $D_0 - D_\infty$, where $D_0 = P_1 + P_2$ is an effective k-rational divisor of degree 2 such that $P_1 \neq \bar{P}_2$. If $D_0 = P_1 + P_2$, generically $P_1, P_2 \notin \{\infty^+, \infty^-\}$, so in this chapter we will only consider generic divisors.

A generic divisor class has a representative $D_0 - D_\infty$ where $D_0 = P_1 + P_2$ with $P_1, P_2 \notin \{\infty^+, \infty^-\}$. Hence, for the remainder of the paper we discuss arithmetic only for generic divisors. This is not a serious restriction for the

pairing applications: there will exist divisor classes of the required prime order which are of the generic form. Full details of how to handle the special cases are given in [36].

Let $D_1 = P_1 + P_2 - D_\infty$ and $D_2 = P_3 + P_4 - D_\infty$ be two divisors. An explicit interpretation of the results of Chapter 3 (see Section 3.2 or Example 3.3.2) in the case of a genus 2 curve implies that if $p(x)$ denotes the unique polynomial of degree at most 3 passing through P_1, P_2, P_3 and P_4, and we let P_5, P_6 be the remaining intersection points of $y - p(x)$ with C, then

$$\mathrm{div}(y - p(x)) = \sum_{i=1}^{6} P_i - 3D_\infty. \tag{5.2.1}$$

If we write $D_3 = \bar{P}_5 + \bar{P}_6 - D_\infty$, Equation (5.2.1) can be rewritten as

$$[D_1] + [D_2] = [D_3].$$

If u_3 is the first polynomial in the Mumford representation of D_3, the function

$$g_{D_1,D_2} = \frac{y - p(x)}{u_3} \tag{5.2.2}$$

has associated divisor $D_1 + D_2 - D_3$. This will be used later to compute pairings (see Equation (2.3.1)).

In our pairing implementation we will use the explicit formulae for Algorithms 3.4,3.5 and 3.6 for genus 2 curves presented in [26], which we include in an Appendix for completeness. The polynomial $p(x)$ in Equation (5.2.1) can be easily computed from the intermediate results in the addition formulae from [26], these calculations are presented in the Appendix.

When the divisor at infinity used is the traditional $D_\infty = 2\infty^+$, the function g_{D_1,D_2} with divisor $D_1 + D_2 - D_3$ has the form $g_{D_1,D_2} = (y - p_1(x))(y - p_2(x))/(u_3(x)u_4(x))$, where again $p_1(x)$ and $p_2(x)$ are cubic polynomials and $u_3(x), u_4(x)$ are quadratic polynomials. Since the bottleneck of pairing calculations is precisely the evaluation of this function, our approach gives a significant improvement over methods which use $D_\infty = 2\infty^+$.

5.2.2 Hyperelliptic curves with embedding degree 6

In this section we will substitute the notation $\mathrm{Cl}^0(C)$ we had been using in previous chapters for the more geometric (and equivalent) $\mathrm{Jac}(C)$, better suited when dealing with endomorphism rings.

In [41], the authors present a family of genus 2 curves with embedding degree 6 and generators of a subring R of the endomorphism ring of $\mathrm{Jac}(C)$, such that R contains a distortion map for any non-trivial pair (D_1, D_2) of divisors.

The curves in this family will have 2 points at infinity and our addition algorithm is well-suited to perform efficient arithmetic on them. We now briefly describe the construction of the curves given in [41, Section 5].

Let $p \neq 2$ a prime such that $p \equiv 2 \pmod 3$. Denote by $\zeta_6 \in \mathbf{F}_{p^2}$ a root of $x^2 - x + 1$ and by $\zeta_3 = \zeta_6^2$, let $\gamma \in \mathbf{F}_{p^6}$ be such that $\gamma^{p^2-1} = \zeta_3$. An equation of C will then be

$$C : y^2 = (ax + b)^6 + (cx + d)^6,$$

where $a = \gamma^p, b = \zeta_3^2\gamma^p, c = \gamma$ and $d = \zeta_3\gamma$.

In this case, the coefficient of the x^6 term in the equation of C is $a^6 + c^6$, which is a non-zero $\mathbf{F}_p$-rational element. If it is not a square, we can take two

rational points on C and move them to the line at infinity, and get a curve isomorphic to C given by a monic polynomial. This will let us use the addition formulae presented in [26], which only work on curves given by an equation of the form $y^2 = x^6 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$.

Lemma 5.2.1. *The model of the curve C defined above has 2 points at infinity.*

Proof. Let C be given by $y^2 = F(x)$ and denote the leading coefficient of F as F_6. Notice that

$$F_6 = a^6 + c^6 = \gamma^{6p} + \gamma^6.$$

To prove the lemma we only need to prove that $F_6 \neq 0$. Since $p^2 - 1$ is a multiple of 3 and $\gamma^{p^2-1} = \zeta_3$ the multiplicative order of γ is a multiple of 9. So $F_6 = \gamma^6(\gamma^{6p-6} + 1)$ cannot be zero as this would imply that $\gamma^{12p-12} = 1$, but $12p - 12$ is not a multiple of 9 as $p \equiv 2 \pmod 3$.

□

The characteristic polynomial of Frobenius on C is $T^4 - pT^2 + p^2$, so $\mathrm{Jac}(C)$ will have $p^2 - p + 1$ elements. Note that if C' is the curve $C' : y^2 = x^6 + 1$, then C is a twist of C' by the automorphism $u : (x, y) \mapsto (\frac{\zeta_3}{x}, \frac{y}{x^3})$. Furthermore, there is an isomorphism $\phi : C \longrightarrow C'$ given by

$$\phi(x, y) = \left(\frac{ax + b}{cx + d}, \frac{y}{(cx + d)^3}\right)$$

The authors of [41] then define the following endomorphisms of C':

$$\pi(x, y) = (x^p, y^p)$$
$$\chi(x, y) = \left(\frac{1}{x}, \frac{y}{x^3}\right)$$
$$\zeta_6(x, y) = (\zeta_6 x, y).$$

We will abuse notation and extend these endomorphisms to $\mathrm{Jac}(C')$. These endomorphisms are enough to find a distortion map on $\mathrm{Jac}(C)$ (see Definition 2.3.7), as the following result shows.

Theorem 5.2.2 (Theorem 5.3 in [41])**.** *Let r be a prime different from 2 and p. Then for all pairs of divisors D_1 and D_2 on C of order r, there exists a distortion map in the ring $\phi^{-1}\mathbb{Z}[\pi, \chi, \zeta_6]\phi$.*

It is well known that if the first coordinate of the Mumford representation of a divisor lies in a proper subfield of $\mathbf{F}_{p^6}$, then the function g_{D_1,D_2} in Equation (5.2.2) can be substituted by $y - p(x)$ (p as in Equation (5.2.2)) in the Miller loop of the pairing computation. The following Lemma shows that the endomorphisms χ and ζ_6 can be used to this end.

Lemma 5.2.3. *Let $P \in C$ be a point with a $\mathbf{F}_p$-rational x-coordinate. Then:*

- *The x-coordinate of $(\phi^{-1} \circ \zeta_6 \circ \phi)(P)$ is $\mathbf{F}_p$-rational.*
- *The x-coordinate of $(\phi^{-1} \circ \chi \circ \phi)(P)$ is $\mathbf{F}_{p^3}$-rational.*
- *The x-coordinate of $(\phi^{-1} \circ \chi \circ \zeta_6 \circ \phi)(P)$ is $\mathbf{F}_{p^3}$-rational.*

Proof. Let $P = (x, y)$ be the coordinates of P. A tedious but simple calculation shows that the x-coordinate of $(\phi^{-1} \circ \zeta_6 \circ \phi)(x, y)$ is given by

$$\frac{-x-1}{x-2},$$

which is $\mathbf{F}_p$-rational whenever x is an element of $\mathbf{F}_p$.

The x-coordinate of $(\phi^{-1} \circ \chi \circ \phi)(x, y)$ is given by

$$x_\chi = \frac{(\zeta_3\gamma^2 - \zeta_3^2\gamma^{2p})x + (\zeta_3^2\gamma^2 - \zeta_3\gamma^{2p})}{(\gamma^{2p} - \gamma^2)x + (\zeta_3^2\gamma^{2p} - \zeta_3\gamma^2)},$$

and again, it is straightforward to prove that $x_\chi^{p^3} = x_\chi$. The third claim follows from the first two. □

The previous Lemma shows that using χ and ζ_6 as distortion maps (see Definition 2.3.7) makes it possible to use *denominator elimination*. We will now prove that the image of $\mathbf{F}_p$-rational divisors under the distortion map $(\phi^{-1} \circ \chi \circ \zeta_6 \circ \phi)$ lies in the p-eigenspace, thus allowing us to directly use loop-shortening techniques.

Theorem 5.2.4. *Let $D_1 \in \mathrm{Cl}^0(C)[r]$ be a $\mathbf{F}_p$-rational divisor. Then its image $D_2 = (\phi^{-1} \circ \chi \circ \zeta_6 \circ \phi)(D_1)$ under the distortion map lies in the p-eigenspace of $\mathrm{Cl}^0(C)[r]$.*

Proof. The r-torsion subgroup $\mathrm{Cl}^0(C)[r]$ can be decomposed as the direct sum of four 1-dimensional eigenspaces with respect to Frobenius π_p, with eigenvalues $1, -1, p$ and $-p$. The polynomial $T^2 - T + 1$ is divisible by $T - p \mod r$, hence the endomorphism $(\pi_p^2 - \pi_p + 1)$ annihilates the p-eigenspace, and is invertible when restricted to the other eigenspaces. It follows that D_2 lies in the p-eigenspace if and only if $(\pi_p^2 - \pi_p + 1)(D_2) = 0$.

To prove that this is the case, it suffices to show that the unique cubic polynomial passing through the four points in the affine support of D_2 and $\pi_p^2(D_2)$ also passes through the points in the affine support of $\pi_p(D_2)$. This can be proven symbolically simply by defining formal variables γ and γ^p over $\mathbb{Q}(\zeta_6)$, and formally defining the action of Frobenius as $\pi_p(\gamma) = \gamma^p$, $\pi_p(\gamma^p) = \zeta_6^2\gamma$ and $\pi_p(\zeta^6) = \zeta_6^5$. The verification of our claim boils down to a trivial, albeit tedious calculation, which we performed using Magma [10]. □

5.2.3 Elliptic curves with embedding degree 3

In this subsection we describe the construction of elliptic curves with embedding degree $k = 3$ given in [95]. We will report our pairing implementation results on these curves in later sections.

Let p be a prime, $p \equiv 5 \bmod 6$, let E be an elliptic curve defined over $\mathbf{F}_{p^2}$ by $y^2 = x^3 + \rho^2$, where $\rho \in \mathbf{F}_{p^2}$ is an element such that ρ^2 is not a cube in $\mathbf{F}_{p^2}$. The number of $\mathbf{F}_{p^2}$ rational points of E is $p^2 - p + 1$ (see Lemma 7 of [42] for a proof). Let r be the largest prime dividing $p^2 - p + 1$, then E has embedding degree $k = 3$ with respect to r. Define the following map:

$$\begin{aligned} \phi_E : E(\mathbf{F}_{p^2}) &\rightarrow E(\mathbf{F}_{p^6}) \\ (x, y) &\rightarrow (a\beta x^p, by^p) \end{aligned}$$

where $a = \rho^{-(2p-1)/3}$, $b = \rho^{-(p-1)}$, and β is a cubic root of ρ. If we let $(x', y') = \phi_E(x, y)$, it is not hard to see that $x' \in \mathbf{F}_{p^6}$ and $y' \in \mathbf{F}_{p^2}$. The endomorphism ϕ_E will be used as a distortion map in our pairing implementation (see Definition 2.3.7).

When executing Miller's algorithm to compute pairings on an elliptic curve, the denominator of the function $g_{n_1,n_2,D}$ in Equation (2.3.1) has the form $(x_R - x_Q)$, where R and Q are points on the elliptic curve. Note that $x_R \in \mathbf{F}_{p^2}$ and $x_Q \in \mathbf{F}_{p^6}$. We replace

$$\frac{1}{x_R - x_Q} = \frac{x_R(x_R + x_Q) + x_Q^2}{y_R^2 - y_Q^2},$$

and since $y_R^2 - y_Q^2$ lies in the proper subfield $\mathbf{F}_{p^2}$ of $\mathbf{F}_{p^6}$, we can discard its

value as it will become 1 after the final exponentiation.

So the function $g_{n_1,n_2,D}$ in Equation (2.3.1) can be substituted by

$$l_{R,P}(Q) \cdot (x_R(x_R + x_Q) + x_Q^2), \tag{5.2.3}$$

where $l_{R,P}$ denotes the line passing through the points P and R. If x_Q^2 is precomputed then the saving compared with the standard method (i.e., writing the Miller variable f as a numerator and a denominator) is to replace a squaring in $\mathbf{F}_{p^6}$ by a multiplication of an element in $\mathbf{F}_{p^2}$ with an element in $\mathbf{F}_{p^6}$.

5.3 Pairing implementation and efficiency analysis

In this section, we describe some optimizations of the pairing implementation on the hyperelliptic curves given above, including the generation of parameters to shorten the Miller loop, denominator elimination, and the finite field construction.

5.3.1 Loop shortening

We will now describe how the loop-shortening techniques presented in Section 2.3 can be used in the curves constructed in Section 5.2 to efficiently compute pairings.

We know that the elliptic curve E constructed in Subsection 5.2.3 accepts a twist of degree 3, has embedding degree $k = 3$ and, in the notation of Subsection 2.3.3, it has $T = p - 1$. Using Theorem 2.3.10 and Equation

(2.3.2), the function

$$e(P,Q) = f_{p-1,P}(Q)^{(q^k-1)/r}$$

defines a bilinear function on $\mathbf{G}_1 \times \mathbf{G}_2$. Therefore, if P is a $\mathbf{F}_{p^2}$-rational point, both $f_{p-1,P}$ and $f_{r,P}$ define bilinear maps on appropriate subgroups of $E[r]$.

Similarly, if D_1 is a r-torsion, $\mathbf{F}_p$-rational divisor on the curve C defined in Subsection 5.2.2, then Theorem 2.3.11 shows that the function f_{p,D_1} defines a bilinear map on appropriate subgroups of $\mathbf{G}_1 \times \mathbf{G}_2$. The standard Tate pairing shows that the function f_{r,D_1} also defines a bilinear pairing on $\mathrm{Cl}^0(C)[r] \times \mathrm{Cl}^0(C)[r]$.

Theorem 5.3.1. *Let C be an elliptic curve constructed as described in Subsection 5.2.3 and let C a curve as described in Subsection 5.2.2. Denote $B = r$, $A_E = p-1$ and $A_C = p$. Choose integers a and b such that $p = a \cdot r + b$. Then the function $f_{b-1,P}$ defines a bilinear map on $\mathbf{G}_1 \times \mathbf{G}_2 \subset E[r] \times E[r]$ and f_{b,D_1} defines a bilinear map on $\mathbf{G}_1 \times \mathbf{G}_2 \subset \mathrm{Cl}^0(C)[r] \times \mathrm{Cl}^0(C)[r]$.*

Proof. This is a straight-forward application of Theorem 2.3.12 and Remark 2.3.13. □

Choosing an appropriate b could greatly improve the pairing computations, we show how to do this in the following section.

Corollary 5.3.2. *Let C be an elliptic curve over $\mathbf{F}_{p^2}$ or a genus 2 curve over $\mathbf{F}_p$ whose divisor class group has $p^2 - p + 1$ points. Let $r \mid (p^2 - p + 1)$ and write $b = (p-1) \pmod r$ in the elliptic case and $b = p \pmod r$ in the genus 2 case. Then the function*

$$f_{b,D_1}(D_2)^{(p^6-1)/r}$$

defines a non-degenerate bilinear pairing on $\mathbf{G}_1 \times \mathbf{G}_2$.

Proof. We already know that $f_{r,D_1}(D_2)^{(p^6-1)/r}$ and (for $T = p-1$ or p respectively) $f_{T,D_1}(D_2)^{(p^6-1)/r}$ are non-degenerate and bilinear. Write $B = r$ and $A = p-1$ or p and apply Theorem 2.3.12. One gets the pairing

$$f_{a,rD_1}(D_2) \cdot f_{b,D_1}(D_2) \cdot g_{arD_1,bD_1}(D_2).$$

Since $rD_1 = 0$ it follows that one can choose $f_{a,rD_1}(D_2) = 1$ and $g_{arD_1,bD_1}(D_2) = 1$. The result follows. □ □

5.3.2 Efficient generation of parameters

In this subsection, we describe a method to generate parameters for the curves constructed in Subsections 5.2.2 and 5.2.3, which will shorten the Miller loop to half the bit-length of the subgroup order r.

Using Corollary 5.3.2, if $b \equiv p \mod r$, then the functions f_{b,D_1} and $f_{b-1,P}$ give bilinear functions on the appropriate subgroups of $\mathrm{Cl}^0(C)[r]$ and $E[r]$ respectively.

In both cases, considering the current security level (AES 80), r is about the same size of p. Algorithm 5.1 below shows how to choose p, r and b efficiently. As can be seen from the algorithm, b can be chosen to have very low hamming weight and half the bit-length of r.

The following is a set of parameters generated by Algorithm 5.1, using $n = 80$. These are the parameters used in our implementation, which will be described in the following section. The parameters are written in hexadecimal notation.

Example 5.3.1. A set of parameters for AES 80 security

- p =B000000000000000011260000000000000006AEFB

Algorithm 5.1 Parameter Generation

INPUT: Integers n, l_{max}.

OUTPUT: Integers b, r, l_1, l_0 and a prime p such that $r \mid p^2 - p + 1, p \equiv b \mod r$, and $p^2 - p + 1 = r(l_1 p + l_0)$.

1: **repeat**
2: Choose b of size n bits and low hamming weight.
3: Let $r = b^2 - b + 1$.
4: **until** r is prime or nearly prime.
5: **for** l from 1 to l_{max}. **do**
6: let $p = l \cdot r + b$.
7: **if** p is a prime and $p \equiv 11 \mod 12$. **then**
8: Break.
9: **if** $l = l_{max}$, goto step 1.
10: let $l_1 = l$ and $l_0 = l(b-1) + 1$. ($p^2 - p + 1 = r(lp + l(b-1) + 1)$)
11: **return** p, r, b, l_1 and l_0.

- r =1000000000000000018F00000000000000009B79
- b =10000000000000000C8

Remark 5.3.3. Algorithm 5.1 can be generalized to find parameters for many other types of curves. For example, a similar algorithm can be used to generate parameters for supersingular genus 2 curves given by an equation of the form $y^2 = x^5 + a$, where $a \in \mathbf{F}_p^*$, $p \equiv 2, 3 \mod 5$. Ó hÉigeartaigh and M. Scott efficiently implemented pairings on these curves in [77], achieving some of the fastest pairing computations on genus 2 curves. Using a parameter selection algorithm similar to Algorithm 5.1 could further improve their results.

5.3.3 Finite field construction and arithmetic

The following field construction was presented by Hu et al. in [51]. We will use it to perform the final exponentiation in our pairing computations.

We restrict to $p \equiv 3 \mod 4$ so that -1 is not a quadratic residue modulo p. In other words, we require $p \equiv 11 \mod 12$. The finite fields are represented

as follows:

$$\begin{aligned}\mathbf{F}_{p^2} &= \mathbf{F}_p[\alpha]/(\alpha^2+1) = \{u\alpha+v | u,v \in \mathbf{F}_p\} = \{a_1 + a_2\beta^3 | a_1, a_2 \in \mathbf{F}_p\}.\\ \mathbf{F}_{p^6} &= \mathbf{F}_{p^2}[\beta]/(\beta^3-\rho) = \{b_0 + b_1\beta + b_2\beta^2 + b_3\beta^3 + b_4\beta^4 + b_5\beta^5 | b_i \in \mathbf{F}_p\}\\ &= \{c_0 + c_1\beta + c_2\beta^2 | c_i \in \mathbf{F}_{p^2}\}\end{aligned}$$

where $\rho = \alpha + u_0$ and u_0 is a small integer such that $x^3 - \rho$ is irreducible over $\mathbf{F}_{p^2}$.

Let M_i, S_i, and I_i denote the cost of multiplication, squaring, and inversion in $\mathbf{F}_{p^i}$ for $i = 1, 2, 6$ using the above representation. It is standard (see Section 7 of [49]) that $M_2 = 3M_1$ and $I_2 = 2M_1 + 2S_1 + 1I_1$. For purposes of comparison we follow [49] and assume that $M_1 = S_1$, and $1I_1 = 10M_1$. Finally, as explained in [21] one has $S_2 = 2M_1$, $M_6 = 15M_1$ and $S_6 = 11M_1$.

Let $e_{ij} \in \mathbf{F}_p$ be defined by $\beta^{ip} = e_{i0} + e_{i1}\beta + \cdots + e_{i5}\beta^5$ for $1 \leq i \leq 5$. We have that $\beta^{ip} = \beta^{2i}\rho^{i(p-2)/3}$. Since $\beta^3 = \rho$ and $\rho \in \mathbf{F}_{p^2}$, there are at most two non-zero terms in the coefficient vector $(e_{i0}, e_{i1}, \cdots e_{i5})$. Specifically, we have $(e_{30}, e_{31}, \cdots e_{35}) = (2w_0, 0, 0, -1, 0, 0)$. Hence, raising a random element to the pth power is given by

$$(b_0 + b_1\beta + b_2\beta^2 + b_3\beta^3 + b_4\beta^4 + b_5\beta^5)^p = b_0 + \sum_{i=1}^{5} b_i(e_{i0} + e_{i1}\beta + \cdots + e_{i5}\beta^5).$$

This computation costs only $8\mathbf{F}_p-$multiplications (remember w_0 is a small integer).

The final exponentiation is often computed using a base p expansion. In the cases $k = 6$, the final exponentiation can be represented as

$$\frac{p^6-1}{r} = (p^3-1)(p+1)\frac{p^2-p+1}{r} = (p^3-1)(p+1)(l_1p + l_0)$$

where l_1 is small. Thus, the construction above allows for very fast exponentiation.

5.3.4 Optimized pairing computation

The cost of Miller's algorithm to compute pairings is determined by the length of the Miller loop, the cost of the calculations inside the loop, and the final exponentiation. To compute pairings on hyperelliptic genus 2 curves given by a real model, we used the techniques described above to speed up the computation, that is:

- Algorithm 5.1 generates suitable parameters to get a short, low Hamming weight Miller loop.
- Use $D_\infty = \infty^+ + \infty^-$ to represent elements of $\mathrm{Cl}^0(C)$ to get fast addition and a simple Miller function.
- The distortion map $(\phi^{-1} \circ \chi \circ \zeta_6 \circ \phi)$ described in Theorem 5.2.4 allows for denominator elimination while using the R-ate pairing [64] technique.
- The field construction in Subsection 5.3.3 provides the arithmetic for a very efficient final exponentiation.

5.4 Efficiency analysis and implementation results

The optimization techniques described above make the computation of pairings on hyperelliptic genus 2 curves practical and efficient. In this section we

analyse the efficiency of our pairing implementations, and compare it with pairing implementations on elliptic curves with the same group order.

5.4.1 Comparison with elliptic curves with $k = 3$

As mentioned in the introduction, the curves constructed in Subsections 5.2.2 and 5.2.3 have very similar characteristics, so implementation results on the embedding degree 3 elliptic curve provide a useful benchmark to analyse our pairing implementation on hyperelliptic curves given by a real model.

As mentioned before, (the class groups of) both curves have the same number of $\mathbf{F}_p$-rational points, and the *embedding field* for both curves is the same, as is the bandwidth requirement. A point $P = (x, y) \in E(\mathbf{F}_{p^2})$ is represented by 4 elements of $\mathbf{F}_p$, which is the same number of coefficients required to represent a divisor $D = (x + u_1 x + u_0, x^3 + v_1 x + v_0)$. Since the target field is the same, both pairing values can be compressed at the same rate by using the technique of the XTR public key cryptosystem [65].

In the notation of Theorem 2.3.12, we need to calculate $f_{b,D}$. Since b is an integer calculated using Algorithm 5.1, it will have very low Hamming weight and we will only analyse the cost of the doubling steps in the Miller loop.

We are pairing two divisors D_1 and D_2 defined over $\mathbf{F}_p$. We assume that these divisors are generic (as mentioned earlier). We further assume that the Mumford representation of D_2 factors over $\mathbf{F}_p$ and that we know the factorisation. This is a restriction to roughly half the divisor classes. Before computing the pairing we apply the distortion map ψ from Theorem 5.2.4 to map D_2 to a divisor over $\mathbf{F}_{p^6}$. In other words, we know $x_1, x_2 \in \mathbf{F}_{p^3}$ and $y_1, y_2 \in \mathbf{F}_{p^6}$ such that $\psi(D_2) = (x_1, y_1) + (x_2, y_2) - D_\infty$.

The Miller functions are evaluated at (x_1, y_1) and (x_2, y_2) rather than performing a resultant computation. Since D_∞ is defined over $\mathbf{F}_p$ we can omit the evaluation $(y - p(x))(D_\infty)$.

To compare the efficiency of our pairing implementations on elliptic and hyperelliptic curves, we first estimate the cost of each doubling step. We will let f denote the intermediate value in the Miller loop. The update of f is similar to that used in other standard implementations of Miller's algorithm, such as Algorithm 1 in Section 2 of [46], except that the denominator of $g_{n_1 D, n_2 D}$ in Equation (2.3.1) can be removed as described by Equation (5.2.3) in the elliptic curve case, and by Lemma 5.2.3 in the hyperelliptic curve case.

elliptic : $f \leftarrow f^2 \cdot l_{R,P}(Q) \cdot (x_R(x_R + x_Q) + x_Q^2)$ and $R \leftarrow 2R$
hyperelliptic: $f \leftarrow f^2 \cdot (y_1 - p(x_1)) \cdot (y_2 - p(x_2))$ and $D_1 \leftarrow 2D_1$.

Here $l_{R,P}$ is the line through R and P, $D_1 = (x_1, y_1) + (x_2, y_2) - D_\infty$ and $y - p(x)$ is as in Equation (5.2.1). Note that $p(x)$ will be a cubic polynomial with coefficients in $\mathbf{F}_p$.

We use affine coordinates for our implementation. In the elliptic case, doubling a point costs about $1I_2 + 2M_2 + 2S_2$, which makes each doubling step in the Miller loop cost about $1I_1 + 67M_1$ (see the formulae for M_2, S_2 etc in Section 4.2). In the hyperelliptic case, doubling a divisor costs about $1I_1 + 32M_1 = 42M_1$ [36], which makes each doubling step in the Miller loop cost about $101M_1$. There are a total of 84 doubling steps using the parameters given in Example 5.3.1. So the costs of the Miller loops are $6468M_1$ and $8484M_1$ respectively (counting only doubling steps since b has Hamming weight 4).

The final exponentiation step is identical in both cases, and costs about $1621M_1$.

This shows that pairings on real hyperelliptic genus 2 curves with $k = 6$ are competitive to parings on elliptic curves with $k = 3$, though slower.

5.4.2 Theoretical comparison with imaginary hyperelliptic curves with $k = 4$

To complement our efficiency analysis, we will also make an abstract comparison of our implementation results with those reported in [77], using genus 2 hyperelliptic curves with embedding degree $k = 4$. The implementation results in [77] are amongst the best reported in the literature.

In curves with embedding degree $k = 4$, the underlying prime field needs to be 96 bits larger than our implementation to achieve an equivalent level of security. The representation of each divisor will then need 384 more bits.

The estimated cost of a pairing computation on a degenerate divisor reported in Section 4.9 of [77] is of about $162I_1 + 10375M_1 + 645S_1$ (excluding the cost of the final exponentiation). This estimate is a bit slower than the estimate for hyperelliptic pairings considered in this paper. Although, as mentioned in Remark 5.3.3, the use of an algorithm similar to Algorithm 5.1 to find curve parameters could improve the results of [77]. However, we expect that a R-ate pairing on this curve for general divisors will not be faster than our case.

We can see that pairings on hyperelliptic curves given by a real model are competitive with pairings on curves given by an imaginary model, in terms of bandwidth and computation requirements.

5.4.3 Implementation results

This section reports some implementation results. The implementation uses the parameters given in Example 5.3.1. The timings are obtained using the Magma Online Platform [10].

The following table summarizes the results. The first row shows our implementation result for hyperelliptic curves, and the second row shows our implementation result for elliptic curves.

Table 5.1: Efficiency Comparision with an AES 80 Security Level

Curve	size of p	Operation Count	time(ms)
$C(\mathbf{F}_p)$ $k=6$	160	$10105M_1$	21.6
$E(\mathbf{F}_{p^2})$ $k=3$	160	$8089M_1$	15.3

5.5 Conclusion

In this chapter we presented several techniques to speed-up the calculation of pairings on hyperelliptic curves given by a real model. We showed that computing pairings on real genus 2 curves is practical. The implementation results are comparable to existing results in the literature for similar settings. We compared the efficiency of two similar elliptic and hyperelliptic curves, and conclude that pairings on elliptic curves with $k = 3$ require 21% less field multiplications than pairings on real hyperelliptic genus 2 curves with $k = 6$. The timing difference in our implementation was that elliptic curves are 28% faster than genus 2 curves.

5.6 Appendix: Addition Formulae

We now present the formulae from [26], which are explicit formulae for the algorithms presented in Section 3.2 to build an efficient algorithm for divisor arithmetic on hyperelliptic curves with two points at infinity. These formulae require that the curve have model of the form

$$y^2 = x^6 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0.$$

To make the polynomial defining the hyperelliptic curve monic, one takes a random pair of $\mathbf{F}_p$-rational points $(x, \pm y)$ on the curve, moves them to infinity, and absorbs the square root of the leading coefficient into y. Since we are working in large characteristic there is no problem setting $f_5 = 0$.

To be compatible with the divisor representation used in [26] the second polynomial in the Mumford representation is the unique polynomial $v' \equiv v \mod u$ of the form $v' = x^3 + v_1x + v_0$. Notice that v' can be represented only by 2 coefficients even though it has degree 3.

When adding divisors D_1 and D_2, the cubic polynomial $p(x)$ given by Equation (5.2.1) can be calculated as $p(x) = v_2(x) + u_2(x)s(x)$, where $s(x) = s_1x + s_0$ in Algorithm 5.2.

The cubic polynomial from Equation (5.2.1) used in Miller's algorithm when doubling a divisor D is given by $p(x) = v(x) + u(x)s(x)$, where $s(x) = s_1x + s_0$ in Algorithm 5.3.

Proposition 5.6.1. *In the notation of Algorithm 5.2, the cubic polynomial $p(x)$ given by Equation (5.2.1) can be calculated as $p(x) = v_2(x) + u_2(x)s(x)$, where $s(x) = s_1x + s_0$. An analogous result holds for the polynomial $p(x) = v(x) + u(x)s(x)$, where $s(x) = s_1x + s_0$ in Algorithm 5.3.*

Algorithm 5.2 Addition Formulae

INPUT: Divisors $D_1 = \mathrm{div}(u_1, v_1)$ and $D_2 = \mathrm{div}(u_2, v_2)$.

OUTPUT: A divisor $D_3 = \mathrm{div}(u_3, v_3)$ such that $[D_3] = [D_1] + [D_2]$.

1: $z_0 = u_{10} - u_{20}, z_1 = u_{11} - u_{21}$.
2: $z_2 = u_{11} \cdot z_1 - z_0, z_3 = u_{10} \cdot z_1$.
3: $r = z_1 \cdot z_3 - z_0 \cdot z_2$.
4: $w_0 = v_{10} - v_{20}, w_1 = v_{11} - v_{21}$.
5: $s'_1 = w_0 \cdot z_1 - w_1 \cdot z_0, s'_0 = w_0 \cdot z_2 - w_1 \cdot z_3$.
6: $k_2 = f_4 - 2v_{21}$.
7: $r_2 = r^2, \hat{w}_0 = r_2 - (s'_1 + r)^2, \hat{w}_1 = (r \cdot \hat{w}^{-1})$.
8: $\hat{w}_2 = \hat{w}_0 \cdot \hat{w}_1, \hat{w}_3 = r \cdot r_2 \cdot w_1$.
9: $s_1 = s'_1 \cdot \hat{w}_2, s_0 = s'_0 \hat{w}_2$.
10: $\tilde{w}_0 s_0 \cdot u_{20}, \tilde{w}_1 = s_1 \cdot u_{21}, l_2 = s_0 + \tilde{w}_1$.
11: $l_1 = (s_0 + s_1)(u_{21} + u_{20}) - \tilde{w}_1 - \tilde{w}_0, l_0 = \tilde{w}_0$.
12: $m'_3 = \hat{w}_3 \cdot (-s_1 \cdot (s_0 + l_2) - 2s_0)$.
13: $m'_2 = \hat{w}_3 \cdot (k_2 - s_1 \cdot (l_1 + 2v_{21}) - s_0 l_2)$.
14: $u'_1 = m'_3 - u_{11}, u'_0 = m'_2 - u_{10} - u_{11} \cdot u'_1$.
15: $\underline{w}_1 = u'_1 \cdot (s_1 + 2), \underline{w}_0 = u'_0 \cdot (l_2 - \underline{w}_1)$.
16: $v'_1 = (u'_0 + u'_1) \cdot (s_1 + -\underline{w}_1 + l_2) - v_{21} - l_1 - \underline{w}_0 - \underline{w}_1$.
17: $v'_0 = \underline{w}_0 - v_{20} - l_0$.

Algorithm 5.3 Doubling Formulae

INPUT: $D = \mathrm{div}(u, v)$.

1: $w_1 = u_1^2, \tilde{v}_1 = 2(v_1 + w_1 - u_0), \tilde{v}_0 = 2(v_0 + u_0 \cdot u_1)$.
2: $w_2 = u_0 \cdot \tilde{v}_1, w_3 = u_1 \cdot \tilde{v}_1$.
3: $\mathrm{inv}_1 = \tilde{v}_1, \mathrm{inv}_0 = w_3 - \tilde{v}_0$.
4: $r = \tilde{v}_0 \cdot \mathrm{inv}_0 - w_2 \cdot \tilde{v}_1$.
5: $k'_2 = f_4 - 2v_1$
6: $k'_1 = f_3 - 2v_0 - 2k'_2 \cdot u_1$.
7: $k'_0 = f_2 - v_1^2 - k'_1 \cdot u_1 - k'_2(w_1 + 2u_0)$.
8: $s'_1 = \mathrm{inv}_1 \cdot k'_0 - \tilde{v}_0 \cdot k'_1, s'_0 = \mathrm{inv}_0 \cdot k'_0 - w_2 \cdot k'_1$.
9: $r_2 = r^2, \hat{w}_0 = (s'_1 + r)^2 - r_2, \hat{w}_1 = (r \cdot \hat{w}_0)^{-1}$.
10: $\hat{w}_2 = \hat{w}_0 \cdot \hat{w}_1, \hat{w}_3 = r \cdot r_2 \cdot \hat{w}_1$.
11: $s_1 = \hat{w}_2 \cdot s'_1, s_0 = \hat{w}_2 \cdot s'_0$.
12: $u'_1 = 2\hat{w}_3 \cdot ((s_0 - u_1) \cdot s_1 + s_0)$.
13: $u'_0 = \hat{w}_3 \cdot ((s_0 2u_1) \cdot s_0 + \tilde{v}_1 \cdot s_1 - k'_2)$.
14: $z_0 = u'_0 - u_0, z_1 = u'_1 - u_1$.
15: $\underline{w}_0 = z_0 \cdot s_0, \underline{w}_1 = z_1 \cdot s_1$.
16: $v'_1 = 2u'_0 - v_1 + (s_0 + s_1) \cdot (z_0 + z_1) - \underline{w}_0 - \underline{w}_1 - u'_1 \cdot (2u'_1 + \underline{w}_1)$.
17: $v'_0 = \underline{w}_0 - v_0 - u'_0 \cdot (2u'_1 + \underline{w}_1)$.

Proof. We will only prove the result corresponding to Algorithm 5.2, since the result for Algorithm 5.3 can be proven analogously.

By construction the polynomial $s(x)$ has the property that $s(x) \equiv (v_1(x) - v_2(x))/u_2(x) \mod u_1(x)$. Hence, the polynomial $p(x) = v_2(x) + u_2(x)s(x)$ is such that $p(x) \equiv v_1(x) \mod u_1(x)$ and $p(x) \equiv v_2(x) \mod u_2(x)$. Since there is a unique cubic polynomial that satisfies these conditions, we have shown that $p(x)$ is indeed the polynomial from Equation (5.2.1). □

Chapter 6

An Attack on Disguised Elliptic Curves

6.1 Introduction

The use of pairings in cryptography has had a number of important consequences. In [70], Menezes, Okamoto and Vanstone use the Weil pairing to reduce the discrete logarithm problem from the group of points of an elliptic curve $E(\mathbf{F}_q)$ to the multiplicative group of invertible elements of a finite field $\mathbf{F}_{q^n}^*$ for a suitable n. In recent years, pairings for elliptic curves have found more constructive applications (see [78] for a survey), which simply stated depend on the fact that they provide some elliptic curves with a gap Diffie-Hellman group structure: a group in which the decision Diffie-Hellman problem is easy, and yet the computational Diffie-Hellman problem remains hard.

In [19], Dent and Galbraith take this construction one step further and explore the idea of Trapdoor Decisional Diffie-Hellman groups: groups for

which the knowledge of certain trapdoor information is sufficient to efficiently solve the DDH, whereas solving the DDH without the trapdoor information is believed to be hard. In [19] the authors describe two such constructions, both based on elliptic curves. The first one depends on elliptic curves over the ring $\mathbf{Z}/N\mathbf{Z}$ where $N = pq$ is an RSA modulus (we refer the reader to the original paper for further details). The second construction is based on an idea of Frey [30] that consists of "disguising" elliptic curves. In the next section we will give a detailed description of this construction and then we will proceed to describe an attack on it. The results of this chapter have been published as [74]

6.2 Disguising elliptic curves

This proposal consists of taking the Weil restriction of an elliptic curve with respect to $\mathbf{F}_{q^n}/\mathbf{F}_q$ and then transforming the group operation equations using a linear change of variables. We will first explain how to obtain multivariate polynomials describing the group law and then we will describe the blinding procedure using an invertible linear transformation.

Given an $\mathbf{F}_q$-basis $\beta = \{\alpha_i\}_1^n$ of $\mathbf{F}_{q^n}$, every element $x = \sum_{i=1}^n x_i\alpha_i$ of $\mathbf{F}_{q^n}$ can be described as an n-tuple $(x_1, x_2, \ldots, x_n) \in \mathbf{F}_q^n$ with respect to β. We will use the notation x to represent both the field element $x \in \mathbf{F}_{q^n}$ and the n-tuple over $\mathbf{F}_q$.

Lemma 6.2.1. *Fix an $\mathbf{F}_q$-basis β of $\mathbf{F}_{q^n}$. Given a monomial x^iy^j, there exists an n-tuple $(p_k)_{k=1}^n$ of homogeneous polynomials p_k of degree $(i+j)$ in $2n$ variables, such that for two elements x_0 and y_0 of $\mathbf{F}_{q^n}$, the coordinates of $x_0^iy_0^j$ are given by the evaluation of the n-tuple of polynomials in the coordinates of*

x_0 and y_0.

Proof. This is a classic result. The n-tuple of polynomials is obtained expanding the product of the formal n-tuples describing x and y and writing the products of elements of β again as an n-tuple in β. □

Throughout this chapter, E will be an elliptic curve defined over the field $\mathbf{F}_{q^n}$. Points on E will be described by their projective coordinates. When denoting projective points, we will use the notation (x, y, z) instead of the usual $(x : y : z)$, because in the attack we will be interested in the specific representative of the projective class being used.

Proposition 6.2.2. *Let E be an elliptic curve defined over a finite field $\mathbf{F}_{q^n}$. There exists homogeneous polynomials f_x, f_y and f_z in $\mathbf{F}_{q^n}[x_1, y_1, z_1, x_2, y_2, z_2]$, such that the addition $P_3 = P_1 + P_2$ of two points P_1 and P_2 on E is given by*

$$P_3 = (f_x, f_y, f_z)(P_1, P_2).$$

Analogously, there exists a triple of polynomials (g_x, g_y, g_z) that give the point doubling operation.

Proof. This is a standard result for elliptic curves. For a proof see [91][Algorithm III.2.3]. □

Proposition 6.2.3. *If we describe a point on $E(\mathbf{F}_{q^n})$ as a $3n$-tuple of elements of $\mathbf{F}_q$, there exists a $3n$-tuple $(f_i)_{i=1}^{3n}$ of homogeneous polynomials f_i in $6n$ variables, such that the addition of two points P_1 and P_2 on E can be calculated as the evaluation of the $3n$-tuple of polynomials in the coordinates of P_1 and P_2. Analogously, there exists a $3n$-tuple $(g_i)_{i=1}^{3n}$ of polynomials g_i in $3n$ variables, that can be used to double points in E.*

Proof. This is a simple combination of Proposition 6.2.2 and Lemma 6.2.1. □

In order to blind the elliptic curve we will choose some random matrix $U \in GL_{3n}(\mathbf{F}_q)$. This matrix will be part of the private key of the cryptosystem.

Definition 6.2.4. *Given an elliptic curve E, let $(f_i)_{i=1}^{3n}$ be the $3n$-tuple of polynomials describing the addition law as in Proposition 6.2.3. Define the $3n$-tuple of blinded addition polynomials as*

$$\left(\tilde{f}_i(x_1, y_1, z_1, x_2, y_2, z_2)\right)_{i=1}^{3n} = U\left(f_i\left(U^{-1}(x_1, y_1, z_1), U^{-1}(x_2, y_2, z_2)\right)_{i=1}^{3n}\right).$$

We define the $3n$-tuple of blinded doubling polynomials $(\tilde{g}_i)_{i=1}^{3n}$ in a similar fashion.

Definition 6.2.5. *Given a point $P = (x, y, z)$ in $\mathbf{F}_{q^n}^3$, write its coordinates as n-tuples with respect to our basis β. We say that the $3n$-tuple $\tilde{P} = U \cdot P$ is the blinded image of P. Throughout this chapter $\tilde{P}$ will denote the blinded image $U \cdot P$ of the point P.*

Definition 6.2.6. *Given an elliptic curve E, we define its blinded description as the tuple*

$$(E, (\tilde{f}_i)_{i=1}^{3n}, (\tilde{g}_i)_{i=1}^{3n}, \tilde{P}_0),$$

where the $3n$-tuples of polynomials $(\tilde{f}_i)$ and $(\tilde{g}_i)$ are the blinded addition and doubling polynomials of E with respect to some matrix U and the $3n$-tuple $\tilde{P}_0$ is the blinded image of a (secret) point P_0 on E.

In [19] different variants of the scheme are discussed; for instance, it is suggested to take a matrix U mapping the XZ- and Y-spaces onto themselves, both for functionality and implementation convenience. A further variant of

the scheme has a more restrictive public key, consisting only of a blinded point $\tilde{P} = U \cdot P$ and the blinded version of the doubling and "translation by P" formulae, this has the disadvantage that it is not possible to compute arbitrary multiples of a point (see the original paper [19] for the details). Our attack does not apply to this variant.

The goal of disguising an elliptic curve is to construct a trapdoor DDH group. Thus, an attack on the scheme is any algorithm that allows someone in possession of the public key to compute a bilinear pairing on the curve. Under such considerations, to break the scheme one does not need to recover the original blinding matrix U, all that is needed is a matrix U' taking our blinded curve to an $\mathbf{F}_{q^n}$-isomorphic curve. In particular, starting with a different $\mathbf{F}_q$-basis of $\mathbf{F}_{q^n}$ corresponds to conjugating U by an invertible matrix, and is enough to break the scheme.

6.3 The attack

In this section we describe our attack on the disguised curve scheme. The attack is based on some simple observations coupled with standard linear algebra. For some variants of the scheme we are able to completely break the trapdoor.

We first present a general attack that will work on any variant with basic functionality; this attack alone does not recover U, but will greatly reduce the search space. Building upon our first attack, we then show a second attack that completely recovers U in some special cases. This second attack can be seen as a warning against careless implementations.

Throughout this section we will fix an $\mathbf{F}_q$-basis $\{\alpha_i\}_1^n$ of $\mathbf{F}_{q^n}$ and whenever

we speak of the matrix in $\mathrm{GL}_n(\mathbf{F}_q)$ associated with multiplication by $\lambda \in \mathbf{F}_{q^n}$, it will be with respect to this basis. If $P = (x, y, z)$ is a point in $\mathbf{F}_{q^n}^3$ and λ is an element of $\mathbf{F}_{q^n}$, then $[\lambda]$ will denote the matrix in $\mathrm{GL}_{3n}(\mathbf{F}_q)$ corresponding to multiplication by λ in each coordinate.

For future reference, we present the standard addition formulae for curves given by an equation of the form $y^2 = x^3 + Ax + B$, or its projective equivalent $zy^2 = x^3 + Axz^2 + Bz^3$. If $(x_3, y_3, z_3) = (x_1, y_1, z_1) + (x_2, y_2, z_2)$, then

$$(x_3, y_3, z_3) = (f_x, f_y, f_z)(x_1, y_1, z_1, x_2, y_2, z_2),$$

where

$$f_x = z_1 z_2 D N^2 - D^3(x_1 z_2 + x_2 z_1) \tag{6.3.1}$$

$$f_y = N(z_1 z_2 N^2 - D^2 x_1 z_2 - 2D^2 x_2 z_1) + D^3 x_2 z_1 \tag{6.3.2}$$

$$f_z = D^3 z_1 z_2 \tag{6.3.3}$$

$$N = y_1 z_2 - y_2 z_1 \quad \text{and} \quad D = x_1 z_2 - x_2 z_1. \tag{6.3.4}$$

Remark 6.3.1. The triple of polynomials (f_x, f_y, f_z) giving addition formulae for the curve E is not unique. For example, given a homogeneous polynomial $p(x_1, y_1, z_1)$, the triple (pf_x, pf_y, pf_z) can also be used to add points on E.

6.3.1 Attack 1

In this first attack we assume that we know the blinded description $(E, (\tilde{f}_i)_{i=1}^{3n}, (\tilde{g}_i)_{i=1}^{3n}, \tilde{P}_0)$ of an elliptic curve E with respect to an unknown matrix U. We do not assume knowledge of the point P_0, or of the unblinded version of the curve

addition formulae. Note that we can find random blinded points on E simply by computing random multiples of $\tilde{P}_0$.

Definition 6.3.2. *Let $\tilde{P}_1$ and $\tilde{P}_2$ be the blinded image of the points $P_1 = (x_1, y_1, z_1)$ and $P_2 = (x_2, y_2, z_2)$. We say that the $3n$-tuples $\tilde{P}_1$ and $\tilde{P}_2$ are* similar *if there exists λ such that $(x_1, y_1, z_1) = (\lambda x_2, \lambda y_2, \lambda z_2)$. In this case we say that λ is the similarity factor between $\tilde{P}_1$ and $\tilde{P}_2$.*

Algorithm 6.1 Similar $3n$-tuples

INPUT: The blinded description $(E, (\tilde{f}_i)_{i=1}^{3n}, (\tilde{g}_i)_{i=1}^{3n}, \tilde{P}_0)$ of an elliptic curve E.
OUTPUT: Two similar $3n$-tuples $\tilde{P}_1$ and $\tilde{P}_2$.
1: Let $\tilde{P}$ and $\tilde{Q}$ be two random blinded points on E.
2: Let $\tilde{P}_1 := 2(\tilde{P} + \tilde{Q})$.
3: Let $\tilde{P}_2 := 2\tilde{P} + \tilde{Q} + \tilde{Q}$.
4: **return** $\tilde{P}_1, \tilde{P}_2$.

Proposition 6.3.3. *Let $\tilde{P}_1, \tilde{P}_2$ be the output of Algorithm 6.1. Then the tuples $\tilde{P}_1$ and $\tilde{P}_2$ are similar, and the similarity factor is a random element of $\mathbf{F}_{q^n}$.*

Proof. The similarity between $\tilde{P}_1$ and $\tilde{P}_2$ follows from the fact that they are both blinded versions of a representative of the point $2P + 2Q$.

Let (p_1, p_2, p_3) be the triple of polynomials that calculates $2P + Q + Q$ (i.e. $(p_1, p_2, p_3)(P, Q) = 2P + Q + Q$) , and let (q_1, q_2, q_3) be the triple of polynomials that calculates $2(P+Q)$. The polynomials p_1 and q_1 have different degrees and one is not a multiple of the other. The proportionality constant by which P_1 and P_2 differ is $(p_1/q_1)(P, Q)$. This proves that it is given by the value of a non-constant rational function evaluated in two random points P and Q. Since non-constant rational functions are surjective over the algebraically closure, we can expect the proportionality constant between P_1 and P_2 to be a random element of $\mathbf{F}_{q^n}$.

The polynomials in the coordinates of P and Q that calculate $2P+Q+Q$ and $2(P+Q)$ have different degrees and one is not a multiple of the other.

The proportionality constant by which P_1 and P_2 differ is the value of

a non-constant rational function evaluated in two random points P and Q on E, there is no reason to expect any constraint in the value by which these two projective points differ when P and Q are taken at random.

□

Proposition 6.3.4. *Fix a triple of polynomials (f_x, f_y, f_z) giving projective addition formulae for the elliptic curve E. There exists a fixed integer s such that if P_1 and P_2 are two different projective representatives of the same point, with coordinates $P_1 = (x_1, y_1, z_1)$ and $P_2 = (\lambda x_1, \lambda y_1, \lambda z_1)$, then for every point Q the projective coordinates of $P_1 + Q$ and $P_2 + Q$are related by*

$$P_1 + Q = (x_3, y_3, z_3), \quad P_2 + Q = (\lambda^s x_3, \lambda^s y_3, \lambda^s z_3).$$

Proof. For any triple of polynomials (f_x, f_y, f_z) giving generic addition formulae on the curve, the polynomials (f_x, f_y, f_z) have to be homogeneous in the coordinates of the first and second points. The integer s is given by the degree of the formulae in the variables corresponding to the first point. □

Proposition 6.3.5. *Let $\tilde{P}_1$ and $\tilde{P}_2$ be two similar $3n$-tuples with similarity factor λ. For a blinded point $\tilde{Q}$, the coordinates of $(\widetilde{P_1 + Q})$ and $(\widetilde{P_2 + Q})$ will differ by the matrix $M = U[\lambda^s]U^{-1}$.*

Proof. If follows directly from the construction of the blinded description of the curve that

$$\tilde{P} + \tilde{Q} = (\widetilde{P + Q}).$$

Using Proposition 6.3.4, we get

$$\tilde{P}_2 + \tilde{Q} = U \cdot (P_2 + Q) = U[\lambda^s] \cdot (P_1 + Q) = U[\lambda^s]U^{-1}(\widetilde{P_1 + Q}) = M(\tilde{P}_1 + \tilde{Q}),$$

which proves the proposition. □

For the attack to succeed we need two similar points $\tilde{P}_1$ and $\tilde{P}_2$ with similarity factor λ such that λ^s generates $\mathbf{F}_{q^n}$. If λ is a random element of $\mathbf{F}_{q^n}$, this happens with high probability as Lemma 6.3.6 shows. If we are unlucky then the attack can be repeated for different pairs of points $(\tilde{P}_1, \tilde{P}_2)$ until we find λ such that λ^s generates $\mathbf{F}_{q^n}$. We will shortly describe how to determine if this is the case.

Lemma 6.3.6. *Let λ be a random element of the field $\mathbf{F}_{q^n}$. The probability that λ^s does not generate $\mathbf{F}_{q^n}$ over $\mathbf{F}_q$ is $O(\log(n)sq^{-n/2})$.*

Proof. The element λ^s does not generate $\mathbf{F}_{q^n}$ if and only if it lies in a proper subfield of $\mathbf{F}_{q^n}$. Every such subfield has at most $q^{n/2}$ elements, and the number of proper subfields of $\mathbf{F}_{q^n}$ that contain $\mathbf{F}_q$ is $O(\log n)$. Finally, if γ is an element of $\mathbf{F}_{q^n}$, there are at most s values of λ such that $\gamma = \lambda^s$. The number of elements λ such that λ^s does not generate $\mathbf{F}_{q^n}$ over $\mathbf{F}_q$ is therefore $O(\log(n)sq^{n/2})$ out of q^n elements of $\mathbf{F}_{q^n}$. The result follows. □

Algorithm 6.2 Similarity Matrix

INPUT: The blinded description of an elliptic curve E, and two similar $3n$-tuples $\tilde{P}_1$ and $\tilde{P}_2$.

OUTPUT: A $3n \times 3n$ matrix M.

1: Construct a set $\{\tilde{Q}_i\}$ of $m > 3n$ random blinded points on E.
2: Let M be a matrix such that $M(\tilde{P}_1 + \tilde{Q}_i) = \tilde{P}_2 + \tilde{Q}_i$ for every $\tilde{Q}_i$.
3: **return** M.

Proposition 6.3.7. *Algorithm 6.2 is correct. If λ is the similarity factor between $\tilde{P}_1$ and $\tilde{P}_2$ and the set $\{\tilde{Q}_i\}$ is sufficiently large, then the unique matrix M satisfying condition 2 in Algorithm 6.2 is given by $M = U[\lambda^s]U^{-1}$.*

Proof. Proposition 6.3.5 shows that $\widetilde{(P_2 + Q_i)} = U[\lambda^s]U^{-1}\widetilde{(P_1 + Q_i)}$ for every Q_i, proving that there is always a matrix satisfying the condition in Step 2 of Algorithm 6.2. If the set of points $\{\tilde{Q}_i\}$ is sufficiently large, there is a unique matrix satisfying these conditions. □

Remark 6.3.8. The eigenvalues of M will be λ^s and its Galois conjugates. We calculate the eigenvalues of M, choose one of them[1] and work with it as λ^s. Using Lemma 6.3.6 we will assume that λ^s generates $\mathbf{F}_{q^n}$ over $\mathbf{F}_q$.

Theorem 6.3.9. *Let $(E, (\tilde{f}_i)_{i=1}^{3n}, (\tilde{g}_i)_{i=1}^{3n}, \tilde{P}_0)$ be the blinded description of an elliptic curve E. Use Algorithm 6.1 to find a pair of similar $3n$-tuples $\tilde{P}_1$ and $\tilde{P}_2$, with similarity factor λ, and use these as input to Algorithm 6.2 to find a matrix M. If U is the secret blinding matrix, then U satisfies*

$$M = U[\lambda^s]U^{-1}. \tag{6.3.5}$$

Proof. This is a simple consequence of Proposition 6.3.7. □

Theorem 6.3.9 imposes a strong condition on the possible matrices U used in the blinding procedure through Equation (6.3.5). There is more than one solution to Equation (6.3.5), so further work has to be done to recover U. Notice that not every matrix U satisfying Equation (6.3.5) can be used as secret key, as its action on points must also be compatible with the point adding and doubling operations.

[1]Choosing the "wrong" λ amounts to twisting the original elliptic curve with some element σ of the Galois group of $\mathbf{F}_{q^n}$ over $\mathbf{F}_q$, this doesn't affect the attack as a useful trapdoor would still be found. Equivalently this can be seen as choosing the $\mathbf{F}_q$-basis $\{\alpha_j^\sigma\}$.

We can rewrite Equation (6.3.5) as

$$MU = U[\lambda^s]. \tag{6.3.6}$$

If we let $\mathcal{V}$ denote the $\mathbf{F}_q$-vector space of matrices U with entries in $\mathbf{F}_q$ satisfying Equation (6.3.6), then every invertible matrix in $\mathcal{V}$ is a solution to Equation (6.3.5). To estimate the strength of the restriction imposed in U by Theorem 6.3.9, we calculate the dimension of the vector space $\mathcal{V}$, and compare it with the number of variables necessary to perform a naive multivariate attack on the scheme. Such an attack would need $9n^2$ variables to represent an arbitrary matrix U in the general case, or $5n^2$ if the XZ- and Y-spaces are mapped onto themselves.

Let $\mathcal{V}_c$ be the vector space of matrices U with entries in the algebraic closure $\overline{\mathbf{F}}_q$ of the field $\mathbf{F}_q$ satisfying Equation (6.3.6).

Proposition 6.3.10. *The dimension of the $\overline{\mathbf{F}}_q$-vector space $\mathcal{V}_c$ is $9n$.*

Proof. We will work over the field $\overline{\mathbf{F}}_q$. Diagonalize the matrices M and $[\lambda^s]$ as $M_D = D_1^{-1} M D_1$ and $M_D = D_2^{-1}[\lambda^s] D_2$. The $\overline{\mathbf{F}}_q$-vector space of matrices V with entries in $\overline{\mathbf{F}}_q$ satisfying

$$M_D V = V M_D, \tag{6.3.7}$$

and the elements of the vector space $\mathcal{V}_c$ are isomorphic as $\overline{\mathbf{F}}_q$-vector spaces: if U is a matrix that satisfies Equation (6.3.6), then the matrix $V = D_1^{-1} U D_2$ satisfies Equation (6.3.7), and conversely if V satisfies Equation (6.3.7), then $U = D_1 V D_2^{-1}$ satisfies Equation (6.3.6).

Since the matrix $[\lambda^s]$ has as eigenvalues all the Galois conjugates of λ^s,

each with multiplicity 3, the matrix M_D has n different values in the diagonal, each repeated 3 times. It is now easy to see that the vector space of matrices V satisfying (6.3.7) has dimension $9n$ as $\overline{\mathbf{F}}_q$-vector space, since it is necessary and sufficient that V maps the 3-dimensional eigenspaces corresponding to a given eigenvalue onto themselves. □

Proposition 6.3.11. *The $\mathbf{F}_q$-vector space $\mathcal{V}$ has dimension $9n$.*

Proof. We know that the $\overline{\mathbf{F}}_q$-vector space $\mathcal{V}_c$ has dimension $9n$, and $\mathcal{V}$ consists of the $\mathbf{F}_q$-rational elements of $\mathcal{V}_c$. The fact that the conditions defining the vector spaces $\mathcal{V}$ and $\mathcal{V}_c$ are defined over $\mathbf{F}_q$ and Proposition A.2.2.10 in [50], prove that there is a basis for the $\overline{\mathbf{F}}_q$-vector space $\mathcal{V}_c$ with elements defined over $\mathbf{F}_q$. The proposition follows. □

Proposition 6.3.12. *If the matrix U maps the XZ- and Y-spaces onto themselves, then the dimension of the vector space $\mathcal{V}$ is $5n$.*

Proof. The proof is analogous to that given for Proposition 6.3.11. □

Given two similar $3n$-tuples $\tilde{P}_1$ and $\tilde{P}_2$, we have seen how Algorithm 6.2 can be used to find a matrix M imposing conditions on the possible blinding matrices U. It would be natural to try to run Algorithm 6.2 using as input several different pairs of similar points $\{\tilde{P}_1', \tilde{P}_2'\}$ to further narrow down the possibilities for U. Unfortunately, this wouldn't give us any extra information, as the following proposition shows.

Proposition 6.3.13. *Let M be the output of Algorithm 6.2 on input $\{\tilde{P}_1, \tilde{P}_2\}$, and let N be the output of Algorithm 6.2 on input $\{\tilde{P}_1', \tilde{P}_2'\}$. Then every matrix U satisfying $M = U[\lambda^s]U^{-1}$ will also satisfy $N = U[\mu^s]U^{-1}$, where μ is the similarity factor between $\tilde{P}_1'$ and $\tilde{P}_2'$.*

Proof. Since λ^s generates $\mathbf{F}_{q^n}$, there exist elements $a_i \in \mathbf{F}_q$ such that $\mu^s = \sum_{i=0}^{n-1} a_i(\lambda^s)^i$. Therefore $N = \sum a_i M^i$, and it follows that for a given matrix U if $U[\lambda^s]U^{-1} = M$, then automatically

$$N = U[\mu^s]U^{-1},$$

so every matrix U satisfying Equation (6.3.5) for $[\lambda^s]$ and M would work for $[\mu^s]$ and N. □

With Theorem 6.3.9, we have reduced the search space for a multivariate attack on the scheme from quadratic in the number of variables n to linear in n (as proven in Proposition 6.3.11 and Proposition 6.3.12). Since Gröbner basis methods can be used to break these systems, our analysis shows that the parameter n required for a given security level would have to be much higher than suggested in [19].

6.3.2 Attack 2

As mentioned before, there are several variants of the disguised curve proposal in [19]. We now show how to improve the previous attack for one of these variants. We will assume knowledge of at least one blinded point in the curve, we also assume that the unblinded version of the addition formulae is given by the polynomials we presented in Equations (6.3.1) above (as we have mentioned, one could use different addition formulae). We will also assume that the XZ (resp. Y)-space is mapped onto itself under U (see [19]) and that $\text{char}(\mathbf{F}_q) > 2$. Since U maps the XZ- and Y-spaces separately, we will write $U = U_{XZ} \oplus U_Y$, where U_{XZ} denotes the action of U on the XZ-space and U_Y gives the action of U on the Y-space.

Algorithm 6.3 Coordinate quotient

INPUT: Two $3n$-tuples $\tilde{A}_1$ and $\tilde{A}_2$ with non-zero Y-coordinates.
OUTPUT: An element μ of $\mathbf{F}_{q^n}$.

1: Let M be the matrix output by Algorithm 6.2. Assume that $M = U[\lambda]U^{-1}$ for some λ that generates $\mathbf{F}_{q^n}$ over $\mathbf{F}_q$, and denote $M = M_{XZ} \oplus M_Y$, where M_{XZ} and M_Y act on the XZ- and Y-spaces respectively.
2: Let $\tilde{A}_{1,Y}$ and $\tilde{A}_{2,Y}$ denote the Y-coordinate of the $3n$-tuples $\tilde{A}_1$ and $\tilde{A}_2$.
3: Find an n-tuple $(a_i)_{i=0}^{n-1}$ of elements of $\mathbf{F}_q$ such that $(\sum_{i=0}^{n-1} a_i M_Y^i)\tilde{A}_{1,Y} = \tilde{A}_{2,Y}$.
4: Let $\mu := \sum_{i=0}^{n-1} a_i \lambda^i$.
5: **return** μ.

Proposition 6.3.14. *In the notation of Algorithm 6.3, let y_1 and y_2 be the unblinded Y-coordinates of the points $\tilde{A}_1$ and $\tilde{A}_2$, and let μ be the value returned by Algorithm 6.3. Then $\mu = y_2/y_1$.*

Proof. Let $y_2/y_1 = \sum_{i=0}^{n-1} a_i \lambda^i$, this implies that $y_2 = (\sum_{i=0}^{n-1} a_i [\lambda]^i) y_1$. We know that $M_Y = U_Y^{-1}[\lambda]U_Y$, and since $\tilde{A}_{1,Y} = U_Y y_1$ and $\tilde{A}_{2,Y} = U_Y y_2$, we get $(\sum_{i=0}^{n-1} a_i M_Y^i)\tilde{A}_{1,Y} = \tilde{A}_{2,Y}$. This proves the proposition. □

Definition 6.3.15. *Let $\mathcal{V}_z$ be the vector space $U \cdot \{(x, y, 0) | x, y \in \mathbf{F}_{q^n}\}$.*

Since the XZ-space and the Y-space are scrambled onto themselves under U, finding a basis for the vector space $\mathcal{V}_z$ is equivalent to finding a basis for the vector space $U\{(x, 0, 0) | x \in \mathbf{F}_{q^n}\}$.

Proposition 6.3.16. *The $3n$-tuple returned by Algorithm 6.4 lies in the vector space $\mathcal{V}_z$.*

Proof. We will use the notation of Algorithm 6.4. The $3n$-tuple $\tilde{A}_3$ is the blinded version of a point $A_3 = (x_3, y_3, z_3)$. This point is also given as $A_3 = A_1 + A_2$, where the "addition" is performed using the unblinded addition formulae given by Equations (6.3.1). Analogously, $\tilde{A}'_3 = UA'_3$ for a point $A'_3 = (x'_3, y'_3, z'_3)$, given by $A'_3 = A'_1 + A_2$.

Algorithm 6.4 Identify $\mathcal{V}_z$

INPUT: Two random $3n$-tuples $\tilde{A}_1$ and $\tilde{A}_2$, corresponding to the blinded representation of the vectors $A_1 = (x_1, y_1, z_1)$ and $A_2 = (x_2, y_2, z_2)$ in $\mathbf{F}_{q^n}^3$.

OUTPUT: A $3n$-tuple $\tilde{A}_4$ lying in the vector space $\mathcal{V}_z$.

1: Let $A_1' = (2x_1, y_1, 2z_1)$, and let $\tilde{A}_1' = UA_1'$. The $3n$-tuple $\tilde{A}_1'$ can be computed from $\tilde{A}_1$ even if A_1' is not known, since U maps the XZ-space onto itself.

2: Denote $\tilde{A}_3 := \tilde{A}_1 + \tilde{A}_2$, and $\tilde{A}_3' := \tilde{A}_1' + \tilde{A}_2$, where the addition is calculated using the blinded addition polynomials.

3: Let $\tilde{A}_4 := 8\tilde{A}_3 - \tilde{A}_3'$ (the subtraction is as $3n$-tuples, not as blinded points of an elliptic curve).

4: **return** $\tilde{A}_4$.

A simple analysis of the addition formulae (6.3.1) shows that $8z_3 = z_3'$ and $8x_3 \neq x_3'$. We now have that

$$8\tilde{A}_3 - \tilde{A}_3' = U(8A_3 - A_3) = U(8x_3 - x_3', 8y_3 - y_3', 0).$$

The proposition follows. □

Corollary 6.3.17. *Given the blinded description of an elliptic curve as presented in Definition 6.2.6, it is possible to efficiently compute a $\mathbf{F}_q$-basis of the vector space $\mathcal{V}_z$.*

Proof. It suffices to run Algorithm 6.4 until a basis of $\mathcal{V}_z$ is found. This should happen with high probability after n runs of the algorithm. □

Proposition 6.3.18. *Algorithm 6.5 is correct.*

Proof. We know that the matrix M is given by $M = U^{-1}[\lambda]U$, where λ generates $\mathbf{F}_{q^n}$ over $\mathbf{F}_q$. If z_1 is the Z-coordinate of A_1, z_2 is the Z-coordinate of A_2 and z_2 is not zero, then there is a unique n-tuple $(a_i)_{i=0}^{n-1}$ of elements of $\mathbf{F}_q$ such that $\sum_{i=0}^{n-1} a_i\lambda^i = z_1/z_2$. It follows that if $A_2' = (\sum_{i=0}^{n-1} a_i\lambda^i)A_2$, then the Z-coordinates of A_2' and A_1 are equal. The proposition follows. □

Algorithm 6.5 Same Z-coordinate

INPUT: Two random $3n$-tuples $\tilde{A}_1$ and $\tilde{A}_2$ not lying in $\mathcal{V}_z$. The matrix M returned by Algorithm 6.2.

OUTPUT: A $3n$-tuple $\tilde{A}'_2$ similar to $\tilde{A}_2$, such that the Z-coordinates of the points A_1 and A'_2 are equal.

1: Corollary 6.3.17 shows that it is possible to find a basis for $\mathcal{V}_z$, so one can use this basis to verify that a vector does not lie in $\mathcal{V}_z$.
2: Find the unique n-tuple $(a_i)_{i=0}^{n-1}$ in $\mathbf{F}_q$, such that $(\sum_{i=0}^{n-1} a_i M^i)\tilde{A}_2 - \tilde{A}_1$ lies in $\mathcal{V}_z$.
3: Let $\tilde{A}'_2 := (\sum_{i=0}^{n-1} a_i M^i)\tilde{A}_2$
4: **return** $\tilde{A}'_2$.

Theorem 6.3.19. *Let $\tilde{P}_0$ be the blinded version of a point P_0 with odd order r. If the affine coordinates of P_0 are $P_0 = (x_0, y_0)$, it is possible to efficiently compute x_0.*

Proof. Recall that given a point $P = (x, y)$ on the elliptic curve E, there exist rational functions $r_1(x)$ and $r_2(x)$ such that $[2]P = (r_1(x), yr_2(x))$.

Let $\tilde{P}_1$ be the result of applying the blinded doubling formulae on $\tilde{P}_0$. Using Algorithm 6.5, find a $3n$-tuple $\tilde{P}'_1$ similar to $\tilde{P}_1$, such that the Z-coordinates of P_0 and P'_1 are the same. Let $P_0 = (X_0, Y_0, Z_0)$ and $P'_1 = (X_1, Y_1, Z_0)$.

Using Algorithm 6.3, find the value w of Y_1/Y_0. Denote the affine coordinates of P_0 and P'_1 by $P_0 = (x_0, y_0)$ and $P'_1 = (x_1, y_1)$. Since $P'_1 = [2]P_0$, we have that $y_1/y_0 = r_2(x_0)$. But also $y_1 = Y_1/Z_0$ and $y_0 = Y_0/Z_0$, hence $y_1/y_0 = Y_1/Y_0$. In other words, the value of Y_1/Y_0 is the same as that of $r_2(x_0)$.

Now solve for x satisfying $r_2(x) = w$. There are at most 4 values of x that satisfy this equation. Given a point $P = (x, y)$ such that x satisfies $r_2(x) = w$, the x-coordinates of the points $P + E[2]$ provide all the other solutions to this equation. At most one of the points in the set $P + E[2]$ has odd order, hence there is only one value of x satisfying $r_2(x) = w$ and such that the point (x, y)

on E has order r. Note that the value of y is only defined up to sign, but this does not change the order of (x, y). The theorem follows. □

Algorithm 6.6 Blinding Matrix

INPUT: The blinded description of an elliptic curve with respect to some unknown matrix U that maps the XZ- and Y-spaces onto themselves.

OUTPUT: A matrix $U' = U \cdot [\mu]$ for some $\mu \in \mathbf{F}_{q^n}$.

1: Find the unblinded affine x-coordinate x_1 of a point $\tilde{P}_1$ using Theorem 6.3.19. Arbitrarily choose y_1 such that $(x_1, y_1) \in E$, and assume that $P_1 = (x_1, y_1)$.
2: For $1 \leq i \leq 9n^2$, compute the point $\tilde{P}_i = [i]\tilde{P}_1$ (here all we need is $9n^2$ known multiples of $\tilde{P}_1$, we use consecutive multiples for convenience).
3: Denote the affine coordinates of the point P_i by (x_i, y_i). These are easily computable from (x_1, y_1) for a given i.
4: Using Algorithm 6.5, substitute $\tilde{P}_i$ for a $3n$-tuple similar to $\tilde{P}_i$, having the same z-coordinate as $\tilde{P}_1$.
5: Find the unique matrix U' such that $\tilde{P}_i = U' \cdot (x_i, y_i, 1)$ for every i.
6: **return** U'.

Algorithm 6.6 shows how to recover the matrix U up to multiplication by a block-diagonal matrix $[\mu]$ for $\mu \in \mathbf{F}_{q^n}$. This is enough to break the scheme, since the factor μ is cancelled when the affine coordinates of a given point are considered.

6.4 Conclusions

We have cryptanalysed the hidden pairing scheme of [19] based on disguising an elliptic curve. Our attacks show that to obtain a secure system one would have to massively increase the memory requirements of the public keys in the proposal of [19]. Our results do not apply to the proposal of Frey since [30] does not specify a method to compute the group law on an elliptic curve.

Chapter 7

Cheon's algorithm, pairing inversion and the DLP

7.1 Introduction

Pairing-based cryptography has become one of the most active research areas in public key cryptography. The security of a pairing-based cryptosystem depends on the difficulty of several computational problems, some of them exclusive to the area, such as the pairing inversion problem (see Definition 7.2.1).

Using results of Verheul [95], later extended by Galbraith, Hess and Vercauteren [37], it is well known that if one can solve certain pairing-inversion problems, then it is also possible to solve the computational Diffie-Hellman (DH) problem in a number of groups, including a class of subgroups of finite fields.

In this chapter we find results that relate the difficulty of pairing inversion problems and the discrete logarithm problem (DLP). We begin using the techniques of Boheh and Lipton [8], and Maurer [68], to show that if one can

solve both the FAPI_1 and FAPI_2 problems (see Definition 7.2.1), then there exists a sub-exponential discrete logarithm algorithm in the groups involved.

We also explore the implications of being able to solve only one of the FAPI problems. In this case it is not possible to solve the computational Diffie-Hellman problem, so the previous approach does not apply. We prove that it is still possible to solve the *static Diffie-Hellman* problem, this will let us use algorithms developed by Brown and Gallant [11], and Cheon [15] that solve the discrete logarithm problem using a static Diffie-Hellman oracle.

Instead of presenting his algorithm in the context of the static Diffie-Hellman problem, Cheon presents his algorithm as a solution to the **l-Strong Diffie-Hellman** problem (l-SDH).

Problem 7.1. Given P and $\alpha^i P$ for $i = 1 \dots l$, compute $\alpha^{l+1} P$.

This problem was first introduced by Boneh and Boyen in [6] to give a security proof in the standard model for a signature scheme. Cheon's idea consists of exploiting the extra information given by the SDH problem to accelerate the computation of the discrete logarithm α.

This chapter is organized as follows. In Section 7.2, we define the pairing inversion problems we are interested in. Section 7.3 uses the techniques developed to reduce the DLP to the DH problem to show that the existence of pairing inversion algorithms implies the existence of sub-exponential discrete logarithm algorithms. Section 7.4 presents Cheon's algorithm and explores its implications in the presence of a pairing inversion oracle. We present our conclusions in Section 7.5

7.2 Pairings

Throughout this chapter, we will let $\mathbf{G}_1, \mathbf{G}_2$ and $\mathbf{G}_T$ denote groups of prime order p. We will write the group operation in $\mathbf{G}_1$ and $\mathbf{G}_2$ additively, and we will use multiplicative notation for $\mathbf{G}_T$. We will consider non-degenerate bilinear pairings of the form

$$e : \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mathbf{G}_T.$$

We are interested in the following problems:

Definition 7.2.1. *Let e be a non-degenerate bilinear pairing as above.*
The **Fixed Argument Pairing Inversion 1 ($\mathbf{FAPI}_1$)** *problem is: given $P_1 \in \mathbf{G}_1, z \in \mathbf{G}_T$, find $P_2 \in G_2$ such that $e(P_1, P_2) = z$.*
The **Fixed Argument Pairing Inversion 2 ($\mathbf{FAPI}_2$)** *problem is: given $P_2 \in \mathbf{G}_2, z \in \mathbf{G}_T$, find $P_1 \in G_1$ such that $e(P_1, P_2) = z$.*

Given an instance (P_1, z) of the FAPI_1 problem, we will denote its solution as $P_2 = \mathrm{FAPI}_1(P_1, z)$. Analogously, a solution to an instance (P_2, z) of the FAPI_2 problem will be denoted as $P_1 = \mathrm{FAPI}_2(P_2, z)$.

The existence of efficient algorithms to solve the FAPI_1 and FAPI_2 problems would have profound consequences. Galbraith, Hess and Vercauteren have generalized results of Verheul, and proved in [37] the following:

Theorem 7.2.2. *[Theorem 1 in [37]] Let $e : \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mathbf{G}_T$ be a non-degenerate bilinear pairing on cyclic groups of prime order p. Given access to $FAPI_1$ and $FAPI_2$ oracles, it is possible to solve the computational Diffie-Hellman problem in $\mathbf{G}_1, \mathbf{G}_2$ and $\mathbf{G}_T$ in polynomial time.*

In practice, $\mathbf{G}_i, i \in \{1, 2\}$, will be a subgroup of an elliptic curve E and

e will be the Tate- or Weil-pairing (or a variant thereof). Let the elliptic curve E be defined over the field K, and suppose that K contains the group of pth roots of unity μ_p. If $E[p]$ denotes the p-torsion subgroup of E, and $E[p] \subset E(K)$, the Tate-pairing is a non-degenerate bilinear function

$$\langle \cdot, \cdot \rangle : E[p] \times E(K)/pE(K) \longrightarrow K^*/(K^*)^p.$$

If K is a finite field, it is possible to get a unique element of K as result of the pairing as

$$e(P, Q) = \langle P, Q \rangle^{(\#\mathbf{K}/p)}.$$

This bilinear function is known as the *reduced Tate-pairing.*

Recent developments in pairing computation techniques, prominently the short Miller loops afforded by the *ate*-pairing [49], have raised questions regarding the possibility of solving one of the FAPI problems for some special curves. In the following sections we will explore some consequences of the existence of pairing inversion algorithms. A detailed description of the subtleties and difficulties regarding efficient pairing inversion can be found in [37, 84].

7.3 FAPI, the DH and DLP problems

After the publication of Verheul's results [95, 96] and with the results recently obtained by Galbraith, Hess and Vercauteren in [37], it is widely known that the ability to invert pairings in polynomial time implies that the computational Diffie-Hellman problem can also be solved in polynomial time.

Combining the results of den Boer [18], Boneh and Lipton [8], and Maurer

and Wolf [68], which relate the Diffie-Hellman problem and the discrete logarithm problem, and the reduction from pairing inversion to the Diffie-Hellman problem proved in [95, 37], we will prove that pairing inversion can be used to solve the discrete logarithm problem in sub-exponential time in the order of the groups. These results, although well-known to experts in the field, have not been published and we include them here to provide a reference.

7.3.1 Black Box Fields

A black-box field is an abstract construction introduced in [8]. It is analogous to the extensively studied black-box groups construction.

Definition 7.3.1. *A black-box field is a 5-tuple (p, S, h, F, G), where p is a prime and S is a set with p elements. The functions h, F, G are defined as follows:*

- *The function $h : S \longrightarrow \mathbb{Z}/p\mathbb{Z}$ is a bijection.*
- *The function $F : S \times S \longrightarrow S$ corresponds to addition, that is $h(F(s_1, s_2)) = h(s_1) + h(s_2)$.*
- *The function $G : S \times S \longrightarrow S$ corresponds to multiplication, that is $h(G(s_1, s_2)) = h(s_1) \cdot h(s_2)$.*

Following Boneh and Lipton, given x an element of $\mathbb{Z}/p\mathbb{Z}$, we will write $[x]$ to denote the element s of S such that $h(s) = x$. Note that the given functions suffice to compute field inversion, since $[x^{-1}] = [x^{p-2}]$. It is interesting to observe that there exist an algorithm by Shanks to extract square roots in $\mathbb{Z}/p\mathbb{Z}$ using only operations available in black-box fields [16]. This observation is fundamental in the techniques developed to relate the DH and DL problems.

Definition 7.3.2. *Let (p, S, h, F, G) be a black-box field for some prime p. Denote the map sending x to $[x]$ by $[\cdot]$. The black-box field problem is: given oracles for $F, G, [\cdot]$ and an element $[\alpha] \in S$, find α explicitly.*

The concept of a black-box field is important because being able to solve the computational Diffie-Hellman problem in a group $\mathbf{G}$ of order p, gives us a black box representation of $\mathbb{Z}/p\mathbb{Z}$ by elements of G.

Definition 7.3.3. *Given an instance (P, aP, bP) of the computational Diffie-Hellman problem, we denote its solution as $abP = \mathrm{DH}(P, aP, bP)$.*

Lemma 7.3.4. *Let $\mathbf{G}$ be group with prime order p generated by P. If we denote the group binary operation as $+$ and let*

$$h : \mathbf{G} \longrightarrow \mathbb{Z}/p\mathbb{Z}$$
$$aP \mapsto a,$$

denote a bijection between $\mathbf{G}$ and $\mathbb{Z}/p\mathbb{Z}$, the 5-tuple $(p, \mathbf{G}, h, +, \mathrm{DH}(P, \cdot, \cdot))$ forms a black-box field representation of $\mathbb{Z}/p\mathbb{Z}$.

Proof. Since $(a+b)P = aP + bP$, it follows that $h(aP) + h(bP) = h((a+b)P)$. Analogously, since $abP = \mathrm{DH}(P, aP, bP)$, we have that $h(\mathrm{DH}(P, aP, bP)) = h(aP) \cdot h(bP)$. □

Note that in this construction $[a] = aP$ for $a \in \mathbb{Z}/p\mathbb{Z}$. In this context, the DLP for the group $\mathbf{G}$ becomes the black-box field problem for $(p, \mathbf{G}, h, +, \mathrm{DH}(P, \cdot, \cdot))$.

The reduction from the DH problem to the DLP presented in [8, 69] uses the following idea of Maurer [68] to solve the DLP problem in the group $\mathbf{G}$ generated by P:

1. Find an elliptic curve $E_{A,B}$, defined over $\mathbb{Z}/p\mathbb{Z}$ by the equation $y^2 = x^3 + Ax + B$, with N-smooth order for a suitably small N. Assume that $E_{A,B}(\mathbb{Z}/p\mathbb{Z})$ is generated by Q.

2. Given P and aP in $\mathbf{G}$, use the black-box representation of $\mathbb{Z}/p\mathbb{Z}$ on $\mathbf{G}$ afforded by the DH oracle and Lemma 7.3.4 to find $[y]$ such that $(a, y) \in E_{A,B}$.

3. Since the order of $E_{A,B}$ is N-smooth, use the Pohling-Hellman algorithm to find the discrete logarithm of (a, y) with respect to Q.

4. Recover a using the known coordinates of Q.

The elliptic curve $E_{A,B}$ is known as an *auxiliary group*, and an approach using more general algebraic groups has been explored in [69].

A run of the algorithm to compute discrete logarithms using a Diffie-Hellman oracle thus consist of two parts: firstly, finding an appropriate curve $E_{A,B}$, and secondly, computing the discrete logarithm of (a, y) with respect to Q. The best result in this direction was proven by Boneh and Lipton in [8], and is presented here as Theorem 7.3.7.

Maurer [68] argues that with high probability there is a number in the interval $[p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p}]$ whose largest prime factor is polynomial in $\log p$. Since for every integer n in this interval there is an elliptic curve over $\mathbb{Z}/p\mathbb{Z}$ with n points [20], knowing the equation defining such an elliptic curve would provide a polynomial time algorithm to solve discrete logarithms in groups of order p with access to a DH-oracle. The implications of these result are not clear, since finding n (and hence the elliptic curve) is likely to be exponentially hard.

Incidentally, Muzereau, Smart and Vercauteren have found auxiliary groups with N-smooth order ($2^{20} \leq N \leq 2^{83}$), for most of the NIST elliptic curves [76]. This is, of course, a hardness result for the Diffie-Hellman problem, as there is no reason to expect the DL problem in this curves to be easy.

7.3.2 Black-Box fields and Pairing Inversion

After the construction described in the previous subsection, a pairing inversion algorithm could then be used as a DH-oracle in the reduction of Boneh and Lipton to solve discrete logarithms in the p-torsion subgroup of an elliptic curve and the group of p-roots of unity μ_p in K. This proves that (conditional to some conjectures regarding the number of N-smooth numbers in Hasse-Weil intervals) there is a reduction from the discrete logarithm problem to solving both of FAPI_1 and FAPI_2.

Definition 7.3.5. *Given a natural number n and a real number α, such that $0 \leq \alpha \leq 1$, denote*

$$L_n(\alpha) = \exp((\log n)^{\alpha}(\log\log n)^{1-\alpha}).$$

Conjecture 7.3.6. [Conjecture 2.10 in [67]] A random integer in the interval $(p+1-2\sqrt{p}, p+1+2\sqrt{p})$ is $L_p(\alpha)$ smooth with probability at least $1/L_p(1-\alpha)^{1-\alpha+o(1)}$ for any α.

Assuming Conjecture 7.3.6, Boneh and Lipton prove the following:

Theorem 7.3.7 (Theorem 8 in [8])**.** *Given a group $\mathbf{G}$ of prime order p, and access to a DH oracle for $\mathbf{G}$, it is possible to compute discrete logarithms in $\mathbf{G}$ in time $L_p(1/2)^{2+o(1)}$.*

Using Theorem 7.3.7, we are ready to prove the main result of this section.

Theorem 7.3.8. *Consider* $e : \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mathbf{G}_T$ *a non-degenerate bilinear pairing. Given access to FAPI*$_1$ *and FAPI*$_2$ *oracles, it is possible to solve the DLP in* $\mathbf{G}_1, \mathbf{G}_2$ *and* $\mathbf{G}_T$ *in time* $L_p(1/2)^{2+o(1)}$

Proof. Using Theorem 7.2.2, the FAPI oracles can be used to construct a Diffie-Hellman oracle for all the groups involved. The result follows immediately from Theorem 7.3.7. □

This theorem proves that the existence of algorithms that efficiently solve the FAPI$_1$ and FAPI$_2$ problems implies the existence of sub-exponential DLP algorithms for the groups involved. However, the Quadratic Sieve and the Number Field Sieve already provide sub-exponential DLP algorithms in finite fields, and using the MOV [70] attack, we get a sub-exponential DLP algorithm for elliptic curves with sufficiently small embedding degree. In this respect, Theorem 7.3.8 is hardly a surprising result.

Furthermore, for a fixed embedding degree k, computing discrete logarithms using our reduction would be slower than a direct attack using the Number Field Sieve on the embedding field $\mathbf{F}_{p^k}$, where discrete logarithms can be computed in time $L_{p^k}(1/3)$. It would be very interesting to find algorithms that accelerate the computation of discrete logarithms using a DH oracle in groups that already have a sub-exponential discrete logarithm algorithm, such as the group of invertible elements in a finite field.

7.4 Cheon's algorithm and the DLP

In the previous section we proved that being able to solve the FAPI problems allows for the computation of discrete logarithms in sub-exponential time.

Note that it might be possible to have algorithms that solve only one of the FAPI problems. In that case, the techniques of Boneh, Lipton, Maurer and Wolf can not be used.

We will prove that one might still adapt an approach developed by Brown and Gallant [11], and Cheon [15], to work in this setting. As we mentioned before, Cheon developed an algorithm to solve the DLP in the context of the l-SDH problem. Using oracle calls to just one FAPI problem, we can use Cheon's algorithm to compute discrete logarithms.

7.4.1 Static Diffie-Hellman Problem

In [11], Brown and Gallant introduce the concept of the *static Diffie-Hellman* (ScDH) problem.

Definition 7.4.1. *Given fixed elements P and aP of the group $\mathbf{G}$, and a random element yP, find ayP.*

Given an instance $((P, aP), Q)$ of the ScDH problem, we will denote its solution as $aQ = \mathrm{ScDH}_{(P,aP)}(Q)$.

The interest in this problem comes from the fact that in many protocols, including static Diffie-Hellman key agreement, a user has a fixed public key aP, and attacks to the system would involve solving an instance of the Diffe-Hellman problem with one of the entries equal to aP. The security of the system from the user's perspective thus depends on the difficulty of solving the ScDH problem and not the traditional DH problem.

Brown and Gallant prove that for a group $\mathbf{G}$ with prime order p, if $p-1 = uv$, and one is given access to a ScDH oracle, it is possible to solve the DLP in $\mathbf{G}$ in time $O(\sqrt{u} + \sqrt{v})$. In the next subsection we present an algorithm

due to Cheon that is similar to the algorithm presented by Brown and Gallant proving their result.

Our interest in the static Diffie-Hellman problem arises from the fact that having access to an oracle that solves exactly one of the FAPI_1 or FAPI_2 problems provides us with a ScDH oracle when interpreted in the appropriate groups.

Proposition 7.4.2. *Given a bilinear pairing $e : \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mathbf{G}_T$, elements $P_1 \in \mathbf{G}_1$ $P_2, \alpha P_2 \in \mathbf{G}_2$ and access to a $FAPI_2$ oracle, it is possible to solve the ScDH problem in $\mathbf{G}_T$ with static input (z, z^α), where $z = e(P_1, P_2)$, in polynomial time.*

Proof. Let $z = e(P_1, P_2)$. Then $z^\alpha = e(P_1, \alpha P_2)$. Given z^y, we can use the FAPI_2 oracle to find $yP_1 = FAPI_2(P_2, z^y)$. We finish simply computing $z^{y\alpha} = e(yP_1, \alpha P_2)$. □

7.4.2 Cheon's algorithm

We now explore Cheon's algorithm [15] and analyse how can it be combined with a FAPI oracle to solve the DLP. We decided to present Cheon's algorithm instead of the very similar solution presented by Brown and Gallant [11], since Cheon's work allows for the computation of discrete logarithms using the factors of either $p - 1$ or $p + 1$, and is more general from our perspective.

Theorem 7.4.3. *[Theorem 1 in [15]] Let P be an element of prime order p in an abelian group. Suppose that d is a positive divisor of $p - 1$. If $P, P_1 = \alpha P$ and $P_d = \alpha^d P$ are given, α can be computed in $O(\log p(\sqrt{(p-1)/d} + \sqrt{d}))$ group operations using $O(\max\{\sqrt{(p-1)/d}, \sqrt{d}\})$ memory.*

To prove Theorem 7.4.3 it suffices to show that Algorithm 7.1 is correct and finishes in the indicated time. The running time of $O(\log p(\sqrt{p/d}+\sqrt{d}))$ for Algorithm 7.1 was later improved to $O(\sqrt{p/d}+\sqrt{d})$ by Kozaki et al in [62].

Algorithm 7.1 Cheon's Algorithm

INPUT: A tuple $(P, P_1 = \alpha P, P_d = \alpha^d P)$, where $d|p-1$.
OUTPUT: α

1: Find a generator $\zeta_0 \in (\mathbb{Z}/p\mathbb{Z})^*$.
2: $\zeta := \zeta_0^d$.
3: $d_1 := \lceil \sqrt{(p-1)/d} \rceil$.
4: Find $0 \leq u_1, v_1 < d_1$ such that $\zeta^{-u_1} P_d = \zeta^{d_1 v_1} P$ by BSGS.
5: $k_0 := d_1 v_1 + u_1$. Note $\alpha^d = \zeta^{k_0}$.
6: $d_2 := \lceil \sqrt{d} \rceil$.
7: Find $0 \leq u_2, v_2 < d_2$ such that $\zeta_0^{-u_2(p-1)/d} P_1 = \zeta_0^{k_0 + d_2 v_2 (p-1)/d} P$ by BSGS.
8: **return** $\zeta_0^{k_0 + (d_2 v_2 + u_2)(p-1)/d}$

Cheon presents several other algorithms to find the discrete logarithm of an element using extra information presented in some cryptographic schemes. Using a special implementation of Algorithm 7.1, Cheon proves the following:

Corollary 7.4.4. *Let P be an element of prime order p in an abelian group. Suppose that $p-1 = \prod_{i=1}^{t} d_i$, for d_i pairwise relatively prime. If P and $P_i = \alpha^{(p-1)/d_i} P$ for $1 \leq i \leq t$ are given, then α can be computed in $O\left(\log p \left(\sum_{i=1}^{t} \sqrt{d_i}\right)\right)$ group operations using $\max\{\sqrt{d_i}\}_{1 \leq i \leq t}$ memory.*

Finally, Cheon represents elements of $\mathbb{F}_{p^2}$ as pairs of elements of $\mathbb{Z}/p\mathbb{Z}$, and uses a clever representation of elements in the subgroup μ_{p+1} of $\mathbb{F}_{p^2}^*$ to prove:

Theorem 7.4.5 (Theorem 2 in [15])**.** *Let P be an element of prime order p in an abelian group. Suppose that d is a positive divisor of $p+1$ and $P_i = \alpha^i P$ for $i = 1, 2, \ldots 2d$ are given. Then α can be computed in $O(\log p(\sqrt{(p+1)/d}+d))$ group operations using $O(\max\{\sqrt{(p+1)/d}, \sqrt{d}\})$ memory.*

7.4.3 Cheon's algorithm and FAPI

Algorithm 7.1 was presented in the context of problems similar to the l-SDH problem as discussed above. However, there is another area where they can potentially be used to attack a cryptosystem.

Proposition 7.4.6. *Given $P_1 \in \mathbf{G}_1$, $P_2, \alpha P_2 \in \mathbf{G}_2$, and access to a $FAPI_2$ oracle, then it is possible to compute $\alpha^i P_1$ for every i using $O(i)$ calls to the $FAPI_2$ oracle.*

Proof. The proof follows from a simple induction argument. Having found $\alpha^n P_1$, we can compute

$$\alpha^{n+1} P_1 = FAPI_2(P_2, e(\alpha^n P_1, P_2).$$

An analogous computation recovers $\alpha^{n-1} P_1$.

□

Using Cheon's algorithm and the previous Proposition, we get the following:

Theorem 7.4.7. *Consider $e : \mathbf{G}_1 \times \mathbf{G}_2 \longrightarrow \mathbf{G}_T$ a non-degenerate bilinear pairing between groups of prime order p. Given $P_2, \alpha P_2 \in \mathbf{G}_2$, access to a $FAPI_2$ oracle and a positive integer d dividing $p-1$ or $p+1$, there exists an algorithm that computes α in time $O(\sqrt{p/d} + d)$.*

Proof. Use d calls to the FAPI_2 oracle to compute $\alpha^d P_2$ as described in Proposition 7.4.6. Given $P_2, \alpha P_2$ and $\alpha^d P_2$ we can use the algorithm in [62] to recover α running in time $O(\sqrt{p/d} + \sqrt{d})$. The result follows. □

Using heuristic results describing the divisors of $p+1$ and $p-1$, we can give an effective version of Theorem 7.4.7. If we assume that for a prime number

p, the prime decomposition of $p+1$ and $p-1$ is the same as that of a random integer, we get the following.

Conjecture 7.4.8 (Section 3 of [11]). The largest prime factor of $p-1$ and $p+1$ is typically of size $O(p^{2/3})$.

Combining Theorem 7.4.7 with Conjecture 7.4.8, we can prove:

Corollary 7.4.9. *Under the hypotheses of Theorem 7.4.7, if p is a random prime, with high probability there exists an algorithm to find α in time $O(p^{1/3})$.*

Proof. If either of $p+1$ or $p-1$ has a prime factor of size $O(p^{2/3})$, then it also has a divisor of size $O(p^{1/3})$. Using this divisor as d in Theorem 7.4.7 gives a running time of $O(p^{1/3})$. □

Note that Pollard's rho method [82] has a running time of $O(n^{1/2})$ to compute discrete logarithms. If we are in a situation where Pollard's rho in $\mathbf{G}_1$ is balanced with the cost of the Number Field Sieve in $\mathbf{G}_T$, Theorem 7.4.7 and Corollary 7.4.9 provide an actual speed-up in the computation of discrete logarithms.

7.5 Conclusions

The relation between pairing inversion algorithms and other well-studied computational problems has only recently received widespread attention [37, 84]. In many pairing-based cryptosystems, the groups $\mathbf{G}_1$ and $\mathbf{G}_2$ are the same, or there is an efficiently computable morphism between them. In these cases, Theorem 7.3.8 proves that if the DLP is hard, no efficient pairing inversion algorithm exists. The same can be argued from Theorem 7.4.7, although in this case the reduction is much looser. As mentioned, if the embedding degree

is fixed, the MOV attack [70] already provides a faster sub-exponential reduction. For some values of k, the MOV attack becomes exponentially slow while pairings can still be computed in polynomial time.

The families of pairing friendly elliptic curves for which the authors of [37] prove that the Miller inversion problem can be solved in polynomial time have embedding degree

$$k \approx \alpha \left(\frac{\log r}{\log \log r} \right),$$

so *if* one could invert pairings for these families, the reduction given by Theorem 7.3.7 would be asymptotically faster than that provided by the MOV attack. Note that the curves in this family are ordinary elliptic curves, so there is no obvious non-trivial morphism between $\mathbf{G}_1$ and $\mathbf{G}_2$.

From a practical point of view, if a sub-exponential but expensive pairing inversion algorithm existed, Theorem 7.4.7 might provide a faster tool to compute discrete logarithms even in cases where an efficiently computable map $\Psi : \mathbf{G}_1 \longrightarrow \mathbf{G}_2$ exists. This is because the algorithms described in Section 7.3 use significantly more calls to a DH oracle in order to compute discrete logarithms than those based in Cheon's algorithm.

If an efficiently computable isomorphism between $\mathbf{G}_1$ and $\mathbf{G}_2$ is known, it is possible to find $\alpha^d P$ using only $O(\log d)$ applications of the FAPI_2 algorithm combining Proposition 7.4.6 and a square-and-multiply algorithm; this would allow us to compute discrete logarithms in $O(\sqrt{p/d} + \sqrt{d} + C \log p)$ operations, where C is the cost of a run of the FAPI_2 algorithm. For example, if either of $p+1$ or $p-1$ has a divisor of size $O(p^{1/2})$, discrete logarithms can be found in $O(p^{1/4} + C \log p)$ operations. Depending on the value of C, this could speed-up the computation of discrete logarithms.

Our results show that the existence of efficient algorithms to solve the

FAPI problems would accelerate the computation of discrete logarithms on some elliptic curves. Depending on the parameters being used, our reduction from the FAPI problems to the DLP might be faster than the reduction given by the MOV attack on pairing friendly elliptic curves.

Bibliography

[1] Adleman, L. M., DeMarrais, J., and Huang, M.-D. A. A subexponential algorithm for discrete logarithms over hyperelliptic curves of large genus over gf(q). *Theor. Comput. Sci. 226*, 1-2 (1999), 7–18.

[2] Adleman, L. M., and Huang, M.-D. Counting points on curves and abelian varieties over finite fields. *J. Symbolic Comput. 32*, 3 (2001), 171–189.

[3] Barreto, P. S. L. M., Galbraith, S. D., Ó hÉigeartaigh, C., and Scott, M. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography 42*, 3 (2007), 239–271.

[4] Barreto, P. S. L. M., and Naehrig, M. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography* (2005), B. Preneel and S. E. Tavares, Eds., vol. 3897 of *Lecture Notes in Computer Science*, Springer, pp. 319–331.

[5] Blake, I. F., Seroussi, G., and Smart, N. P., Eds. *Advances in elliptic curve cryptography*, vol. 317 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 2005.

[6] Boneh, D., and Boyen, X. Short signatures without random oracles. In *EUROCRYPT* (2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 56–73.

[7] Boneh, D., and Franklin, M. K. Identity-based encryption from the weil pairing. In *CRYPTO* (2001), J. Kilian, Ed., vol. 2139 of *Lecture Notes in Computer Science*, Springer, pp. 213–229.

[8] Boneh, D., and Lipton, R. J. Algorithms for black-box fields and their application to cryptography (extended abstract). In *CRYPTO* (1996), N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, Springer, pp. 283–297.

[9] Bosma, W., Ed. *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings* (2000), vol. 1838 of *Lecture Notes in Computer Science*, Springer.

[10] Bosma, W., Cannon, J., and Playoust, C. The Magma algebra system. I. The user language. *J. Symbolic Comput. 24*, 3-4 (1997), 235–265. Computational algebra and number theory (London, 1993).

[11] Brown, D. R. L., and Gallant, R. P. The static diffie-hellman problem. Cryptology ePrint Archive, Report 2004/306, 2004.

[12] Buchmann, J., and Williams, H. C. A key exchange system based on real quadratic fields. In *CRYPTO* (1989), G. Brassard, Ed., vol. 435 of *Lecture Notes in Computer Science*, Springer, pp. 335–343.

[13] Cantor, D. G. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp. 48*, 177 (1987), 95–101.

[14] Cassels, J. W. S., and Flynn, E. V. *Prolegomena to a middlebrow arithmetic of curves of genus* 2, vol. 230 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, Cambridge, 1996.

[15] Cheon, J. H. Security analysis of the strong diffie-hellman problem. In *EUROCRYPT* (2006), S. Vaudenay, Ed., vol. 4004 of *Lecture Notes in Computer Science*, Springer, pp. 1–11.

[16] Cohen, H. *A course in computational algebraic number theory*, vol. 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993.

[17] Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., and Vercauteren, F., Eds. *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete Mathematics and its Applications (Boca Raton). Chapman & Hall/CRC, Boca Raton, FL, 2006.

[18] den Boer, B. Diffie-hellman is as strong as discrete log for certain primes. In *CRYPTO* (1988), S. Goldwasser, Ed., vol. 403 of *Lecture Notes in Computer Science*, Springer, pp. 530–539.

[19] Dent, A. W., and Galbraith, S. D. Hidden pairings and trapdoor ddh groups. In Hess et al. [48], pp. 436–451.

[20] Deuring, M. Die Typen der Multiplikatorenringe elliptischer Funktionenkörper. *Abh. Math. Sem. Hansischen Univ. 14* (1941), 197–272.

[21] Devegili, A. J., Ó hÉigeartaigh, C., Scott, M., and Dahab, R. Multiplication and squaring on pairing-friendly fields, 2006. Cryptology ePrint Archive: Report 2006/471.

[22] Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Trans. Information Theory IT-22*, 6 (1976), 644–654.

[23] Dupont, R., Enge, A., and Morain, F. Building curves with arbitrary small MOV degree over finite prime fields. *J. Cryptology 18*, 2 (2005), 79–89.

[24] Duursma, I. M., and Lee, H.-S. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In *ASIACRYPT* (2003), C.-S. Laih, Ed., vol. 2894 of *Lecture Notes in Computer Science*, Springer, pp. 111–123.

[25] Eisenträger, K., and Lauter, K. A CRT algorithm for constructing genus 2 curves over finite fields. *Algebraic Geometry and Coding Theory 2007* (2007). To appear.

[26] Erickson, S., Jacobson, M. J., Shang, N., Shen, S., and Stein, A. Explicit formulas for real hyperelliptic curves of genus 2 in affine representation. In *WAIFI* (2007), C. Carlet and B. Sunar, Eds., vol. 4547 of *Lecture Notes in Computer Science*, Springer, pp. 202–218.

[27] Freeman, D. Constructing pairing-friendly genus 2 curves with ordinary jacobians. In Takagi et al. [93], pp. 152–176.

[28] Freeman, D., Scott, M., and Teske, E. A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372, 2006.

[29] Freeman, D., Stevenhagen, P., and Streng, M. Abelian varieties with prescribed embedding degree. In van der Poorten and Stein [94], pp. 60–73.

[30] Frey, G. How to disguise an elliptic curve (weil descent).

[31] Frey, G., and Lange, T. Fast bilinear maps from the tate-lichtenbaum pairing on hyperelliptic curves. In Hess et al. [48], pp. 466–479.

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

[33] Fulton, W. *Algebraic curves*. Advanced Book Classics. Addison-Wesley Publishing Company Advanced Book Program, Redwood City, CA, 1989. An introduction to algebraic geometry, Notes written with the collaboration of Richard Weiss, Reprint of 1969 original.

[34] Galbraith, S., Paterson, K., and Smart, N. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006.

[35] Galbraith, S. D. Supersingular curves in cryptography. In *ASIACRYPT* (2001), C. Boyd, Ed., vol. 2248 of *Lecture Notes in Computer Science*, Springer, pp. 495–513.

[36] Galbraith, S. D., Harrison, M., and Mireles Morales, D. J. Efficient hyperelliptic arithmetic using balanced representation for divisors. In van der Poorten and Stein [94], pp. 342–356.

[37] Galbraith, S. D., Hess, F., and Vercauteren, F. Aspects of pairing inversion. Cryptology ePrint Archive, Report 2007/256, 2007.

[38] Galbraith, S. D., Hess, F., and Vercauteren, F. Hyperelliptic pairings. In Takagi et al. [93], pp. 108–131.

[39] Galbraith, S. D., Lin, X., and Mireles Morales, D. J. Pairings on hyperelliptic curves with a real model. *Proceedings of the Pairing 2008 conference* (2008). To appear.

[40] Galbraith, S. D., McKee, J., and Valença, P. Ordinary abelian varieties having small embedding degree. Cryptology ePrint Archive, Report 2004/365, 2004.

[41] Galbraith, S. D., Pujolas, J., Ritzenthaler, C., and Smith, B. Distortion maps for genus two curves.

[42] Galbraith, S. D., and Verheul, E. R. An analysis of the vector decomposition problem. In *Public Key Cryptography* (2008), R. Cramer, Ed., vol. 4939 of *Lecture Notes in Computer Science*, Springer, pp. 308–327.

[43] Gaudry, P. An algorithm for solving the discrete log problem on hyperelliptic curves. In *EUROCRYPT* (2000), pp. 19–34.

[44] Gaudry, P., and Harley, R. Counting points on hyperelliptic curves over finite fields. In Bosma [9], pp. 313–332.

[45] Gaudry, P., Houtmann, T., Kohel, D., Ritzenthaler, C., and Weng, A. The 2-adic cm method for genus 2 curves with application to cryptography. In *ASIACRYPT* (2006), X. Lai and K. Chen, Eds., vol. 4284 of *Lecture Notes in Computer Science*, Springer, pp. 114–129.

[46] Granger, R., Hess, F., Oyono, R., Thériault, N., and Vercauteren, F. Ate pairing on hyperelliptic curves. In *EUROCRYPT* (2007), M. Naor, Ed., vol. 4515 of *Lecture Notes in Computer Science*, Springer, pp. 430–447.

[47] Hartshorne, R. *Algebraic geometry*. Springer-Verlag, New York, 1977. Graduate Texts in Mathematics, No. 52.

[48] Hess, F., Pauli, S., and Pohst, M. E., Eds. *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings* (2006), vol. 4076 of *Lecture Notes in Computer Science*, Springer.

[49] Hess, F., Smart, N. P., and Vercauteren, F. The eta pairing revisited. *IEEE Transactions on Information Theory 52*, 10 (2006), 4595–4602.

[50] Hindry, M., and Silverman, J. H. *Diophantine geometry*, vol. 201 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2000. An introduction.

[51] Hu, L., Dong, J.-W., and Pei, D. Implementation of cryptosystems based on tate pairing. *J. Comput. Sci. Technol. 20*, 2 (2005), 264–269.

[52] Huang, M.-D., and Ierardi, D. Counting points on curves over finite fields. *J. Symbolic Comput. 25*, 1 (1998), 1–21.

[53] Igusa, J.-i. Arithmetic variety of moduli for genus two. *Ann. of Math. (2) 72* (1960), 612–649.

[54] Igusa, J.-i. On Siegel modular forms of genus two. *Amer. J. Math. 84* (1962), 175–200.

[55] Jacobson, M., Scheidler, R., and Stein, A. Fast arithmetic on hyperelliptic curves via continued fraction expansions. In *Advances in Coding Theory and Cryptography* (2007), T. Shaska, W. Huffman, D. Joyner,

and V. Ustimenko, Eds., vol. 3 of *Series on Coding Theory and Cryptology*, World Scientific Publishing, pp. 201–244.

[56] Jacobson, M. J., Scheidler, R., and Stein, A. Cryptographic protocols on real hyperelliptic curves. *Adv. Math. Commun. 1*, 2 (2007), 197–221.

[57] Joux, A. A one round protocol for tripartite diffie-hellman. In Bosma [9], pp. 385–394.

[58] Katagi, M., Akishita, T., Kitamura, I., and Takagi, T. Some improved algorithms for hyperelliptic curve cryptosystems using degenerate divisors. In *ICISC* (2004), C. Park and S. Chee, Eds., vol. 3506 of *Lecture Notes in Computer Science*, Springer, pp. 296–312.

[59] Katagi, M., Kitamura, I., Akishita, T., and Takagi, T. Novel efficient implementations of hyperelliptic curve cryptosystems using degenerate divisors. In *WISA* (2004), C. H. Lim and M. Yung, Eds., vol. 3325 of *Lecture Notes in Computer Science*, Springer, pp. 345–359.

[60] Kedlaya, K. S. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. *J. Ramanujan Math. Soc. 16*, 4 (2001), 323–338.

[61] Koblitz, N. Hyperelliptic cryptosystems. *J. Cryptology 1*, 3 (1989), 139–150.

[62] Kozaki, S., Kutsuma, T., and Matsuo, K. Remarks on cheon's algorithms for pairing-related problems. In Takagi et al. [93], pp. 302–316.

[63] Lange, T. Formulae for arithmetic on genus 2 hyperelliptic curves. *Appl. Algebra Engrg. Comm. Comput. 15*, 5 (2005), 295–328.

[64] Lee, E., Lee, H.-S., and Park, C.-M. Efficient and generalized pairing computation on abelian varieties. Cryptology ePrint Archive, Report 2008/040, 2008.

[65] Lenstra, A. K., and Verheul, E. R. The XTR public key system. In *CRYPTO* (2000), M. Bellare, Ed., vol. 1880 of *Lecture Notes in Computer Science*, Springer, pp. 1–19.

[66] Lenstra, Jr., H. W. On the calculation of regulators and class numbers of quadratic fields. In *Number theory days, 1980 (Exeter, 1980)*, vol. 56 of *London Math. Soc. Lecture Note Ser.* Cambridge Univ. Press, Cambridge, 1982, pp. 123–150.

[67] Lenstra, Jr., H. W. Factoring integers with elliptic curves. *Ann. of Math. (2) 126*, 3 (1987), 649–673.

[68] Maurer, U. M. Towards the equivalence of breaking the diffie-hellman protocol and computing discrete algorithms. In *CRYPTO* (1994), Y. Desmedt, Ed., vol. 839 of *Lecture Notes in Computer Science*, Springer, pp. 271–281.

[69] Maurer, U. M., and Wolf, S. The diffie-hellman protocol. *Des. Codes Cryptography 19*, 2/3 (2000), 147–171.

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

[71] Mestre, J.-F. Construction de courbes de genre 2 à partir de leurs modules. In *Effective methods in algebraic geometry (Castiglioncello, 1990)*, vol. 94 of *Progr. Math.* Birkhäuser Boston, Boston, MA, 1991, pp. 313–334.

[72] Mireles Morales, D. J. An analysis of the infrastructure in real function fields. Submitted for publication.

[73] Mireles Morales, D. J. Cheon's algorithm, pairing inversion and the discrete logarithm problem. Submitted for publication.

[74] Mireles Morales, D. J. An attack on disguised elliptic curves. *Journal of Mathematical Cryptology 2* (2008), 1–8.

[75] Miyaji, A., Nakabayashi, M., and Takano, S. Characterization of elliptic curve traces under fr-reduction. In *ICISC* (2000), D. Won, Ed., vol. 2015 of *Lecture Notes in Computer Science*, Springer, pp. 90–108.

[76] Muzereau, A., Smart, N. P., and Vercauteren, F. The equivalence between the DHP and DLP for elliptic curves used in practical applications. *LMS J. Comput. Math. 7* (2004), 50–72 (electronic).

[77] Ó hÉigeartaigh, C., and Scott, M. Pairing calculation on supersingular genus 2 curves. In *Selected Areas in Cryptography* (2006), E. Biham and A. M. Youssef, Eds., vol. 4356 of *Lecture Notes in Computer Science*, Springer, pp. 302–316.

[78] Paterson, K. G. Cryptography from pairings. In *Advances in elliptic curve cryptography*, vol. 317 of *London Math. Soc. Lecture Note Ser.* Cambridge Univ. Press, Cambridge, 2005, pp. 215–251.

[79] Paulus, S., and Rück, H.-G. Real and imaginary quadratic representations of hyperelliptic function fields. *Math. Comp. 68*, 227 (1999), 1233–1241.

[80] Paulus, S., and Stein, A. Comparing real and imaginary arithmetics for divisor class groups of hyperelliptic curves. In *Algorithmic number theory (Portland, OR, 1998)*, vol. 1423 of *Lecture Notes in Comput. Sci.* Springer, Berlin, 1998, pp. 576–591.

[81] Pila, J. Frobenius maps of abelian varieties and finding roots of unity in finite fields. *Math. Comp. 55*, 192 (1990), 745–763.

[82] Pollard, J. M. Monte Carlo methods for index computation (mod p). *Math. Comp. 32*, 143 (1978), 918–924.

[83] Sakai, R., Ohgishi, K., and Kasahara, M. Cryptosystems based on pairing. *Symposium on Cryptography and Information Security (SCIS2000), Okinawa, Japan* (2000).

[84] Satoh, T. On pairing inversion problems. In Takagi et al. [93], pp. 317–328.

[85] Scheidler, R. Cryptography in quadratic function fields. *Des. Codes Cryptography 22*, 3 (2001), 239–264.

[86] Scheidler, R., Stein, A., and Williams, H. C. Key-exchange in real quadratic congruence function fields. *Des. Codes Cryptography 7*, 1-2 (1996), 153–174.

[87] Schoof, R. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp. 44*, 170 (1985), 483–494.

[88] Scott, M., and Barreto, P. S. L. M. Generating more mnt elliptic curves. *Des. Codes Cryptography 38*, 2 (2006), 209–217.

[89] Shamir, A. Identity-based cryptosystems and signature schemes. In *CRYPTO* (1984), G. R. Blakley and D. Chaum, Eds., vol. 196 of *Lecture Notes in Computer Science*, Springer, pp. 47–53.

[90] Shanks, D. The infrastructure of a real quadratic field and its applications. In *Proceedings of the Number Theory Conference (Univ. Colorado, Boulder, Colo., 1972)* (Boulder, Colo., 1972), Univ. Colorado, pp. 217–224.

[91] Silverman, J. H. *The arithmetic of elliptic curves*, vol. 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1992. Corrected reprint of the 1986 original.

[92] Silverman, J. H. *Advanced topics in the arithmetic of elliptic curves*, vol. 151 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1994.

[93] Takagi, T., Okamoto, T., Okamoto, E., and Okamoto, T., Eds. *Pairing-Based Cryptography - Pairing 2007, First International Conference, Tokyo, Japan, July 2-4, 2007, Proceedings* (2007), vol. 4575 of *Lecture Notes in Computer Science*, Springer.

[94] van der Poorten, A. J., and Stein, A., Eds. *Algorithmic Number Theory, 8th International Symposium, ANTS-VIII, Banff, Canada, May 17-22, 2008, Proceedings* (2008), vol. 5011 of *Lecture Notes in Computer Science*, Springer.

[95] Verheul, E. R. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. In *EUROCRYPT* (2001), B. Pfitzmann, Ed., vol. 2045 of *Lecture Notes in Computer Science*, Springer, pp. 195–210.

[96] Verheul, E. R. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology 17*, 4 (2004), 277–296.

[97] Weng, A. Constructing hyperelliptic curves of genus 2 suitable for cryptography. *Math. Comp. 72*, 241 (2003), 435–458 (electronic).