Fast Formulas for Computing Cryptographic Pairings

by

Craig Costello

Bachelor of Applied Science (Mathematics) with Distinction
(Queensland University of Technology) – 2007
Bachelor of Applied Science Honours (Mathematics) - First Class Honours
(Queensland University of Technology) – 2007

Thesis submitted in accordance with the regulations for the Degree of Doctor of Philosophy

Information Security Institute
Faculty of Science and Technology
Queensland University of Technology

November 25, 2012

Keywords

Tate pairing, ate pairing, explicit formulas, elliptic curves, Weierstrass curves, Miller's algorithm, precomputation, twists, pairing-friendly curves, subfamilies, pairing implementation, high-security pairings, hyperelliptic curves, group law, Jacobian arithmetic, genus 2.

Abstract

The most powerful known primitive in public-key cryptography is undoubtedly elliptic curve pairings. Upon their introduction just over ten years ago the computation of pairings was far too slow for them to be considered a practical option. This resulted in a vast amount of research from many mathematicians and computer scientists around the globe aiming to improve this computation speed. From the use of modern results in algebraic and arithmetic geometry to the application of foundational number theory that dates back to the days of Gauss and Euler, cryptographic pairings have since experienced a great deal of improvement. As a result, what was an extremely expensive computation that took several minutes is now a high-speed operation that takes less than a millisecond.

This thesis presents a range of optimisations to the state-of-the-art in cryptographic pairing computation. Both through extending prior techniques, and introducing several novel ideas of our own, our work has contributed to record-breaking pairing implementations.

For Mum.

Contents

List of Figures

List of Tables

List of Algorithms

Symbols and abbreviations

(f)	divisor of the function f
$[n]P$	scalar multiplication (exponentiation) of P by $n \in \mathbb{Z}$
$\#E$	number of points on E
$\mathbb{A}^n(K)$	affine n-space over the field K
$\ell_{R,Q}$	indeterminate Miller line function in the addition of R and Q
$\ell_{R,R}$	indeterminate Miller line function in the doubling of R
$\epsilon(D)$	effective part of the divisor D
η_T	eta (T) pairing
$\mathbb{F}_q$	finite field with q elements
$\mathbb{F}_{q^k}$	full extension field
$\mathbb{G}_1$	base field subgroup: $E[r] \cap \ker(\pi - [1])$ (in Type 3 pairing)
$\mathbb{G}_2$	trace-zero subgroup: $E[r] \cap \ker(\pi - [q])$ (in Type 3 pairing)
$\mathbb{G}_T$	order r subgroup of $\mathbb{F}_{q^k}^*$ (commonly the r-th roots of unity μ_r)
$\mathrm{Jac}(C_g)$	Jacobian of genus g hyperelliptic curve
$\hat{\mathrm{J}}\mathrm{ac}(C_g)$	dense set of full degree divisor classes on Jacobian of genus g hyperelliptic curve
$K_{\mathrm{ADD}}^{\mathrm{Mum}}$	Mumford function field for addition
$K_{\mathrm{DBL}}^{\mathrm{Mum}}$	Mumford function field for doubling

$\mathcal{O}$	point at infinity on an elliptic curve E
$\overline{K}$	algebraic closure of the field K
$\mathbb{P}^n(K)$	projective n-space over the field K
ϕ	occurs as the distortion map on supersingular curves and as the GLV endomorphism
Φ_i	i-th cyclotomic polynomial
π	q-power Frobenius endomorphism: $(x, y) \mapsto (x^q, y^q)$
Ψ	the (un)twisting isomorphism
ψ	occurs as both the isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ and as the GLS isomorphism
$\psi_\ell(x)$	ℓ-th division polynomial on E (for odd ℓ)
ρ	ratio between base field size and subgroup size for a pairing-friendly curve
a_T	ate pairing
C	an arbitrary curve
C_g	(imaginary quadratic) hyperelliptic curve of genus g
D	occurs as both a divisor on E and the CM discriminant of E
d	degree of twist
D_P	divisor $(P) - (\mathcal{O})$
D_Q	divisor $(Q) - (\mathcal{O})$
E	an elliptic curve
e	a general pairing
E'	twisted curve (defined over $\mathbb{F}_{q^{k/d}}$)
$E(K)$	set of K-rational points on E

$e(P,Q)$	pairing of P and Q (the paired value)
E/K	elliptic curve defined over K
$E[r]$	the (entire) r-torsion
$f_{m,P}$	function with divisor $(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O})$
g	genus of a curve
K	arbitrary field
k	embedding degree of E (with respect to q and r)
n_P	multiplicity of point P in associated divisor
$N_{L/K}(\alpha)$	norm of $\alpha \in L$ over K
P	generator of $\mathbb{G}_1$
P'	twist of the point P
Q	generator of $\mathbb{G}_2$
Q'	twist of the point Q
r	order of the large prime subgroup in $E(\mathbb{F}_q)$
T	ate pairing loop parameter ($T = t - 1$)
t	trace of Frobenius
$T_r(P,Q)$	order r reduced Tate pairing
$t_r(P,Q)$	order r Tate pairing
$w_r(P,Q)$	order r Weil pairing
aTr	anti-trace map
BKLS − GHS	Barreto-Kim-Lynn-Scott/Galbraith-Harrison-Soldera algorithm
BLS	Barreto-Lynn-Scott families
BN	Barreto-Naehrig family with $k = 12$

CM	complex multiplication
$\mathrm{Deg}(D)$	degree of the divisor D
$\mathrm{Div}^0(E)$	group of degree zero divisors on E
$\mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$	group of divisors on $E/\mathbb{F}_q$
DLP	discrete logarithm problem
ECC	elliptic curve cryptography
ECDLP	elliptic curve discrete logarithm problem
$\mathrm{End}(E)$	endomorphism ring of E
$\mathrm{Gal}(L/K)$	Galois group of L over K
GLS	Galbraith-Lin-Scott method
GLV	Gallant-Lambert-Vanstone method
HECC	hyperelliptic curve cryptography
KSS	Kachisa-Schaefer-Scott families
MNT	Miyaji-Nakabayashi-Takano (construction/criteria)
NIST	National Institute of Standards and Technology
NSS	not supersingular curves
$\mathrm{ord}_P(f)$	the multiplicity of f at P on E
PBC	pairing-based cryptography
$\mathrm{Pic}^0(E)$	Picard group of E
$\mathrm{Prin}(E)$	group of principal divisors on E
$\mathrm{QR}(q)$	set of quadratic residues modulo q
$\mathrm{supp}(D)$	support of the divisor D
Tr	trace map

Declaration

The work contained in this thesis has not been previously submitted for a degree or diploma at any higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed:  **Date:** 04/01/13

Previously Published Material

The following papers have been published or presented, and contain material based on the content of this thesis.

Craig Costello, Huseyin Hisil, Colin Boyd, Juan Manuel Gonzlez Nieto, and Kenneth Koon-Ho Wong. Faster pairings on special Weierstrass curves. In Hovav Shacham and Brent Waters, editors, *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 89-101. Springer, 2009.

Craig Costello, Tanja Lange, and Michael Naehrig. Faster pairing computations on curves with high-degree twists. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 224-242. Springer, 2010.

Craig Costello, Colin Boyd, Juan Manuel Gonzlez Nieto, and Kenneth Koon-Ho Wong. Avoiding full extension field arithmetic in pairing computations. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 203-224. Springer, 2010.

Craig Costello, Colin Boyd, Juan Manuel Gonzlez Nieto, and Kenneth Koon-Ho Wong. Delaying mismatched field multiplications in pairing computations. In M. Anwar Hasan and Tor Helleseth, editors, *WAIFI*, volume 6087 of *Lecture Notes in Computer Science*, pages 196-214. Springer, 2010.

Craig Costello and Douglas Stebila. Fixed argument pairings. In Michel Abdalla and Paulo S. L. M. Barreto, editors, *LATINCRYPT*, volume 6212 of *Lecture Notes in Computer Science*, pages 92-108. Springer, 2010.

Craig Costello, Kristin Lauter and Michael Naehrig. Attractive subfamilies of

BLS curves for implementing high-security pairings. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *INDOCRYPT*, volume 7107 of *Lecture Notes in Computer Science*, pages 320-342. Springer, 2011.

Craig Costello and Kristin Lauter. Group law computations on Jacobians of hyperelliptic curves. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 92-117. Springer, 2011.

Craig Costello. Particularly friendly members of family trees. Cryptology ePrint Archive, Report 2012/072, 2012. `http://eprint.iacr.org/`. *In submission.*

Acknowledgements

The last four years have been the best time of my life. I have been extremely spoilt on many accounts, for which I have many people to thank.

I am greatly indebted to my supervisors Colin Boyd and Juan Manuel González Nieto. They helped steer the ship early on, but trusted me to take the reigns down the track. Their support, advice and friendship has been a huge help. I also give thanks to Kenneth Koon-Ho Wong, who was my associate supervisor in the early days and who taught me Magma.

I owe a great deal to my mentor, collaborator and friend Huseyin Hisil. He essentially gifted me the results in our first paper which kick-started my research and, in doing so, he showed me the ropes.

I have been so lucky to meet many of my academic idols and even more fortunate to work with some of them. For a small fish in such a large pond, these discussions, collaborations and friendships have meant a great deal to me.

I sincerely thank Tanja Lange for her encouragement and collaboration, and thank both her and Dan Bernstein for visiting us here at QUT.

I sincerely thank Alice Silverberg and the Number Theory group at the University of California - Irvine for having me at UCI throughout my Fulbright year.

I am extremely grateful to Kristin Lauter for her quality mentorship and support, for having me as an intern at Microsoft Research, for her enthusiasm in discussions, and for being so generous with her broad and deep insight.

I am very grateful to Michael Naehrig for his mentorship, collaboration and for being able to prove things when I am stuck.

I sincerely thank Mike Scott for his all-seeing eyes in the field of pairings and for his generous feedback upon reading this thesis. I further thank him, Naomi Benger and Luis J. Dominguez Perez for their hospitality in my brief stay at DCU in Dublin.

I sincerely thank Paulo Barreto for his motivating encouragement, discussions and friendship... and for the caipirinhas in São Paulo. I also thank Paulo for his comments on this thesis.

I thank Karst Koymans for carefully reading this thesis and providing many valuable comments.

I thank my friend and colleague Douglas Stebila, *withoot* whom there would be no Canadian accents in the office to poke fun at.

I thank all of my fellow co-authors for their collaboration and contributions to the work in this thesis: Colin, Juanma, Kenneth, Huseyin, Douglas, Tanja, Michael and Kristin.

I am extremely grateful for the financial support and funding invested in me and in this work. I sincerely thank the Australian Research Council for an APA scholarship, QUT for top-up scholarships and travel money, the Queensland Government for the Smart State bursary, and the Australian-American Fulbright Commission for the generous grant. I am also very grateful to the many conferences and workshops that have afforded me stipends. In particular, I mention the São Paulo Advanced School of Cryptography in 2011 and ECC2010 in Redmond, both of which were highlight conferences during my candidature.

On a personal note, I am deeply thankful to my tight-knit family for their incredible love and constant support. To my big sister Brit who always looks out for me and has my back; to my brother Ben who calls me every day when I'm abroad; to my brother John who forgets that I am abroad when he calls; and to my brother Clay who constantly reminds me that none of this work will ever excuse my inability to change a car tyre.

I sincerely thank Ruth and Phil for their generosity, love and support; Maurice and Trish for their love and generous hospitality; and Hannah and Mitch for being such fun roommates.

I am deeply grateful to my best friend and girlfriend Nicole, who has been by my side and shared every step of this journey. From delivering pizzas to my office at midnight when I was cramming for my first submission, to conference-hopping around Europe and living with me in the US, all the way up to drawing Figures 2.22 - Figures 2.33 that saved me many painstaking hours in LaTeX.

Lastly, this thesis is dedicated to my mum, who is my hero and to whom no words are profound enough. Without her unfathomable love and belief in me, none of this would have been possible.

Chapter 1

Introduction

When explaining the magic of *public-key cryptography* to the layman, one is usually met with true astonishment. Indeed, Diffie and Hellman achieved the seemingly impossible in 1976 [DH76] by showing that two parties can establish a shared secret over an insecure channel without any prior knowledge of one another. Layman or otherwise, one cannot gain any intuition towards their groundbreaking result without being aware of the existence of *one-way functions*. A one-way function is one that is easy to compute in one direction, but which is believed infeasible to compute in the other. Diffie and Hellman used the simple example of exponentiation in finite fields to introduce the notion of public-key cryptography. Specifically, they showed that a party A can bury their secret value a by raising some publicly known generator g to the power of a in $\mathbb{F}_q$, i.e. computing $g_a = g^a \in \mathbb{F}_q$. The value g_a (called A's public key) can then be sent over an insecure channel to another party B, whose secret value is b, to which B can exponentiate g_a and compute $g_{ab} = g_a^b$. Conversely, b can compute their public key $g_b = g^b$ and send it over an insecure channel to A, who can compute the shared secret $g_{ab} = g_b^a$. In this case the individual secrets of both parties are assumed to be safe under the *discrete logarithm problem* (DLP) in $\mathbb{F}_q$, which is defined as follows: given $g \in \mathbb{F}_q$ and $h = g^x \in \mathbb{F}_q$, find x. If q is large enough, so that solving the DLP is infeasible (with all the computing power on the planet), then exponentiation is the one-way function that keeps a and b out of reach for eavesdroppers and attackers. Additionally, we assume that computing $g_{ab} = g^{ab}$ is infeasible for adversaries that know g, g_a and g_b; this is the *Diffie-Hellman*

problem in $\mathbb{F}_q$.

Discrete logarithm based protocols can be achieved in any groups where the corresponding DLP is hard to solve. Almost a decade after its invention, the abstraction of public-key cryptography beyond traditional groups like $\mathbb{F}_q$ was independently proposed in the pioneering works of Miller [Mil85] and Koblitz [Kob87], who both suggested using elliptic curve groups. This gave birth to the now thriving field of elliptic curve cryptography (ECC), which has gained huge popularity because the discrete logarithm problem on elliptic curves is believed to be much harder than that in finite fields. Consequently, ECC facilitates the use of much smaller key sizes, which not only reduces storage but gives drastic improvements in efficiency. Not long after elliptic curves were suggested, Koblitz took this abstraction even further by proposing the more general setting which employs the Jacobian group of a hyperelliptic curve [Kob89].

It was not until the turn of the century however, that the cryptographic community witnessed the most remarkable feature offered by elliptic curves and their relatives from the domain of algebraic and arithmetic geometry. Namely, these objects facilitate a primitive that is much more powerful than the standard discrete logarithm primitives – a cryptographic *pairing*. The seminal works of Joux [Jou04], Sakai, Ohgishi and Kasahara [SOK00] and Boneh and Franklin [BF03] triggered an avalanche of novel and exciting protocols which exploit the powerful bilinearity property of pairings. Hundreds of papers followed, covering vast amounts of new cryptographic territory that is practically unattainable without pairings. The most notable achievement being the realisation of practical identity-based encryption which, starting with the original solution due to Boneh and Franklin, has since experienced a great deal of generalisations and extensions. Well known commercial companies, such as Hewlett-Packard, have taken a lot of interest in the new cryptographic technology while new companies, such as Voltage Security Inc., have been founded on the idea of pairings. For a survey of protocols in pairing-based cryptography (PBC), and of the problems on which these protocols are based, we refer the reader to Paterson's chapter [Pat05].

These landmark achievements have placed a great emphasis on the efficient computation of cryptographic pairings. Indeed, upon their introduction just over a decade ago, the computation of pairings was too slow for any of the above protocols to be of real practical use. This created a flurry of research from mathematicians and computer scientists around the globe that aimed to improve

the computation speed of pairings. As a result, what was an extremely expensive operation that took several minutes is now a high-speed computation that takes less than a millisecond.

In this thesis we present a range of improvements to the state-of-the-art in pairing implementation. Both through extending prior techniques, and introducing several novel ideas of our own, our work has contributed to record-breaking implementations.

Our contributions

In Chapter 2 we present an extensive survey of the arena of cryptographic pairing computation. We specifically target readers with limited background that are new to the field, for which we hope our exposition provides a comprehensive but friendly first read.

The derivation and optimisation of explicit formulas is one of the major research gaps addressed in this thesis. Chapter 3 is based on our joint work with Huseyin Hisil, Colin Boyd, Juan Manuel Gonzalez Nieto and Kenneth Koon-Ho Wong [CHB+09], and on our joint work with Tanja Lange and Michael Naehrig [CLN10]. We derive record-breaking projective formulas for computations within the pairing algorithm that are tailor-made for all cases of practical interest. Whilst the work in [CHB+09] was not originally applied to the *ate pairing*, a theorem we give in [CLN10] has allowed us to revisit the results from [CHB+09] herein, and apply the earlier work to state-of-the-art ate pairing implementations.

The first few sections of Chapter 4 explore the idea of *loop unrolling* in general pairing computations over large prime fields. This is based on the joint works with Colin Boyd, Juan Manuel Gonzalez Nieto and Kenneth Koon-Ho Wong [CBNW10a, CBNW10b]. Although these works achieve speed ups in some cases (the Tate pairing), they do not improve the start-of-the-art (the ate pairing) computations for standalone one-off pairing computations. However, in Section 4.4 we apply the loop unrolling technique to the common scenario of a pairing that contains a *fixed argument*, showing that large speed ups in state-of-the-art implementations can be achieved by employing our technique. This is based on joint work with Douglas Stebila [CS10], but at the time of writing this thesis we noticed a significant improvement to our original work in that makes our method

even more beneficial.

Both Chapter 5 and Chapter 6 are aimed at identifying attractive subfamilies of *pairing-friendly curves*. This is based on our joint work with Kristin Lauter and Michael Naehrig [CLN11], and on our recent follow up work in [Cos12]. Here we give *implementation-friendly* subfamilies for nine of the most popular families of pairing-friendly curves. Our results in both of these chapters lead to highly efficient pairing instantiations at a range of security levels.

In Chapter 7, we develop an alternative to Cantor's algorithm for group law computations on Jacobians of arbitrary hyperelliptic curves. This is based on our joint work with Kristin Lauter [CL11]. Although the results can be applied to pairings, our work in Chapter 7 has a greater impact outside the context of PBC. This is partly due to the fact that hyperelliptic curve pairings are not currently competitive with elliptic curve pairings. Nevertheless, our explicit algorithm can immediately be used to implement hyperelliptic arithmetic on curves of any genus. As one implication, we applied our method to give more efficient formulas for group operations on general genus 2 hyperelliptic curves. Since our algorithm inherently computes the functions required in pairing computation, this work speeds up hyperelliptic pairings as well.

This thesis concludes in Chapter 8 with some brief comments on possible directions for future work.

Chapter 2

Pairings

Aficionados of cryptographic pairing computation are often asked by interested newcomers to point towards literature that is a good starting point. My answer usually differs depending on the mathematical background volunteered from the "pairing beginner", but almost always involves accordingly picking a subset of the following excellent references.

- Galbraith's chapter [Gal05] is a stand-out survey of the field (up until 2005). It provides several theorems and proofs fundamental to pairing-based cryptography and gives some useful toy examples that illustrate key concepts.

- Lynn's thesis [Lyn07] is also a great survey of the entire arena of pairing computation (up until 2007), and gives all the details surrounding the pioneering papers he co-authored [BKLS02, BLS02, BLS03, BLS04], which are themselves good starting points.

- The first chapter of Naehrig's thesis [Nae09, Ch. 1] conveniently presents the necessary algebro-geometric results required to be able to read most of the literature concerning pairing computation.

- Scott's webpage [Sco04] gives a short and very friendly introduction to the basics of the groups involved in pairing computations by means of an illustrative toy example.

- In his new chapter entitled Algorithmic Aspects of Elliptic Curves, Silverman's second edition [Sil09, Ch. XI.7] includes a concise introduction to pairing-based cryptography that also points to foundational results found elsewhere in his book.

In addition, digging up talks from some of the big players in the field is usually (but not always!) a good way to avoid getting bogged down by minor technical details that slow one's progress in grasping the main ideas. In particular, we refer to the nice talks by Scott [Sco07a, Sco07b] and Vercauteren [Ver06b, Ver06a].

In any case, correctly prescribing the best reading route for a beginner naturally requires individual diagnosis that depends on their prior knowledge and technical preparation. A student who is interested in learning pairings, but who has never seen or played with an elliptic curve, may quickly become overwhelmed if directed to dive straight into the chapters of Silverman's book or Naehrig's thesis. This is not due to lack of clarity, or to lack of illuminating examples (both chapters are ample in both), but perhaps more because of the vast amount of technical jargon that is necessary for one to write a complete and self-contained description of cryptographic pairings. On the other hand, an informal, example-driven approach to learning the broad field of pairing computation may ease the beginner's digestion in the initial stages. For instance, a novice would be likely to find it more beneficial to first see the simple toy example of the quadratic twisting isomorphism in action on Scott's webpage [Sco04], before heading to Silverman's book [Sil09, Ch. X.5.4] to see all possible twisting isomorphisms formally defined, and then later returning to his earlier chapters (specifically Ch. II.2) to read about maps between curves in full generality.

In this light we discuss the major aim of this preliminary chapter. We intend to let illustrative examples drive the discussion and present the key concepts of pairing computation with as little machinery as possible. For those that are fresh to pairing-based cryptography, it is our hope that this chapter might be particularly useful as a first read and prelude to more complete or advanced expositions (e.g. the related chapters in [Gal12]). Of course, the ultimate purpose of this chapter is to ensure we collect the prerequisites that form the basis for the novel contributions that follow in the main chapters, but in some cases we may postpone presenting results and pick them up "on-the-fly" later on, especially if this helps to achieve a more beginner-friendly discussion here. On the other hand, we also hope our introduction does not leave any sophisticated readers

dissatisfied by a lack of formality or generality, so in cases where our discussion does sacrifice completeness, we will at least endeavour to point to where a more thorough exposition can be found.

One advantage of writing a survey chapter on pairing computation in 2012 is that, after more than a decade of intense and fast-paced research by mathematicians and cryptographers around the globe, the field is now racing towards full maturity. Therefore, an understanding of this chapter will equip the reader with most of what they need to know in order to tackle any of the vast literature in this remarkable field, at least for a while yet. Anyone who understands our examples will also comfortably absorb the basic language of algebraic geometry in the context of curve-based cryptography. Since we are aiming the discussion at active readers, we have matched every example with a corresponding snippet of (hyperlinked) Magma [BCP97] code[1], where we take inspiration from the helpful Magma pairing tutorial by Dominguez Perez *et al.* [DKS09]. In the later sections of this chapter we build towards a full working pairing code that encompasses most of the high-level optimisations; this culminates to finish the chapter in Example 2.6.11.

We organise this chapter as follows. We start in Section 2.1 by giving an overview of elliptic curve cryptography (ECC). Indeed, elliptic curves are the main object on which cryptographic pairings take place, so this first section forms a basis for the entire chapter. In Section 2.2 we introduce the important concept of divisors, as well as other essential theory from algebraic geometry that is needed to properly understand cryptographic pairings. In Section 2.3 we detail the specific elliptic curve groups that are employed in a cryptographic pairing, before presenting Miller's algorithm to compute the Weil and Tate pairings in Section 2.4. In Section 2.5 we introduce the notion of *pairing-friendly curves* and give a brief survey of the most successful methods of constructing them. In Section 2.6 we bring the reader up to speed with the landmark achievements and improvements that have boosted pairing computation to the point it is today.

[1]If one does not have access to Magma, the scripts we provide can be run at the online Magma calculator: `http://magma.maths.usyd.edu.au/calc/`

2.1 Elliptic curves as cryptographic groups

The purpose of this section is to introduce elliptic curves as they are used in cryptography. Put simply, an elliptic curve is an abstract type of *group*.

Perhaps a newcomer will find this abstractness apparent immediately when we insist that to understand elliptic curve groups in cryptography, the reader should be familiar with the basics of *finite fields* $\mathbb{F}_q$. This is because, more generally, elliptic curves are groups which are defined on top of (over) fields. Even though elliptic curve groups permit only one binary operation (the so called *group law*), the operation itself is computed within the *underlying field*, which by definition permits two operations (and their inverses). For a general field K, the group elements of an elliptic curve E are *points* whose (x, y) coordinates come from $\overline{K}$ (the algebraic closure of K), and which satisfy the (affine) curve equation for E, given as

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.1}$$

where $a_1, ..., a_6 \in \overline{K}$. Equation (2.1) is called the *general Weierstrass equation* for elliptic curves. Aside from all the $(x, y) \in \overline{K}$ solutions to the equation above, there is one extra point which can not be defined using the affine equation, but which must be included to complete the group definition. This point is called the *point at infinity*, which we denote by $\mathcal{O}$, and we will define it properly in a moment.

If $a_1, ..., a_6 \in K$, then we say E is *defined over* K, and write this as E/K (the same goes for any extension field L of K). Before we go any further, we make a convenient simplification of the general Weierstrass equation. If the field characteristic is not 2 or 3, then divisions by 2 and 3 in K permit the substitutions $y \mapsto (y - a_1x - a_3)/2$ to give $E : y^2 = 4x^3 + b_2x^2 + 2b_4x + b_6$, and then $(x, y) \mapsto \left(\frac{x-3b_2}{36}, \frac{y}{108}\right)$, which (upon appropriate rescaling) yields the following simplified equation.

$$E : y^2 = x^3 + ax + b. \tag{2.2}$$

Equation (2.2) is called the *short Weierstrass equation* for elliptic curves, and will be used all the way through this thesis. Namely, we will always be working over large prime fields, where the short Weierstrass equation covers all possible

isomorphism classes of elliptic curves, so the curves we use will always be an instance of (2.2).

Example 2.1.1 (Magma script). $E/\mathbb{Q} : y^2 = x^3 - 2$ is an elliptic curve. Along with the point at infinity $\mathcal{O}$ (which we are still yet to define), the set of points over $\mathbb{Q}$ is written as $E(\mathbb{Q})$, and is defined as $E(\mathbb{Q}) = \{(x, y) \in \mathbb{A}^2(\mathbb{Q}) : y^2 = x^3 - 2\} \cup \{\mathcal{O}\}$. The point $P = (x_P, y_P) = (3, 5)$ lies in $E(\mathbb{Q})$, as do $Q = (x_Q, y_Q) = \left(\frac{129}{100}, \frac{-383}{1000}\right)$ and $R = (x_R, y_R) = \left(\frac{164323}{29241}, \frac{-66234835}{5000211}\right)$, so we can write $P, Q, R \in E(\mathbb{Q})$. We usually write E to represent the group of points over the full algebraic closure, so for example, the point $S = (x_S, y_S) = \left(0, \sqrt{-2}\right) \in E = E(\overline{\mathbb{Q}})$, but $S \notin E(\mathbb{Q})$. Soon we will be defining the binary group operation $\oplus$ on E using rational formulas in the underlying field, so an active reader can return to this example with these formulas to verify that $R = P \oplus Q$, where x_R, y_R are computed from x_P, y_P, x_Q, y_Q using additions and multiplications (also subtractions and inversions) in $\mathbb{Q}$. Furthermore, it can also be verified that $Q = P \oplus P$, so that $R = P \oplus P \oplus P$; we usually write these as $Q = [2]P$ and $R = [3]P$, where $\underbrace{P \oplus P \cdots \oplus P}_{n} = [n]P$ in general. To finish this example, we remark that if $(x', y') \in E$, then $(x', -y') \in E$ (but is not distinct if $y' = 0$), which is true for any elliptic curve in short Weierstrass form.

Example 2.1.2 (Magma script). $E/\mathbb{F}_{11} : y^2 = x^3 + 4x + 3$ is an elliptic curve. $E(\mathbb{F}_{11})$ has 14 points: $(0, 5)$, $(0, 6)$, $(3, 3)$, $(3, 8)$, $(5, 4)$,$(5, 7)$, $(6, 1)$, $(6, 10)$, $(7, 0)$, $(9, 3)$, $(9, 8)$, $(10, 3)$, $(10, 8)$, not forgetting the point at infinity $\mathcal{O}$. Notice that all but two points come in pairs (x', y') and $(x', -y')$, the exceptions being $(x', y') = (7, 0)$ (since $y' = -y' = 0$) and $\mathcal{O}$. If we form the quadratic extension $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$ with $i^2 + 1 = 0$, then considering E over $\mathbb{F}_{q^2}$ will allow many more solutions, and give many more points: namely, $\#E(\mathbb{F}_{q^2}) = 140$. In addition to the points in $E(\mathbb{F}_q)$, $E(\mathbb{F}_{q^2})$ will also contain those points with x-coordinates in $\mathbb{F}_q$ that did not give $x^3 + 4x + 3$ as a quadratic residue in $\mathbb{F}_q$ (but necessarily do in $\mathbb{F}_{q^2}$), and many more with both coordinates in $\mathbb{F}_{q^2} \setminus \mathbb{F}_q$. Examples of both such points are $(2, 5i)$ and $(2i + 10, 7i + 2)$ respectively. It is not a coincidence that $\#E(\mathbb{F}_q) \mid \#E(\mathbb{F}_{q^2})$, since $E(\mathbb{F}_q)$ is a subgroup of $E(\mathbb{F}_{q^2})$.

Not every tuple $(a, b) \in K \times K$ gives rise to the curve given by $f(x, y) = y^2 - (x^3 + ax + b) = 0$ being an elliptic curve. If there exists $P = (x_P, y_P)$ on f such that both partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ vanish simultaneously at P, then P is called a *singular* point and f is also deemed singular. Conversely, if no such point exists, f is called *non-singular*, or *smooth*, and is then an elliptic curve. It

is easy enough to show that a singularity occurs if and only if $4a^3 + 27b^2 = 0$ (see [Sil09, Ch. III.1, Prop. 1.4]), so as long as $4a^3 + 27b^2 \neq 0$ in K, then $E/K : y^2 = x^3 + ax + b$ is an elliptic curve.

In cryptography we only ever instantiate elliptic curves defined over finite fields, but it is often conceptually helpful to view graphs of elliptic curves over $\mathbb{R}$. We illustrate the difference between singular and non-singular (smooth) elliptic curves in Figures 2.1-2.4.

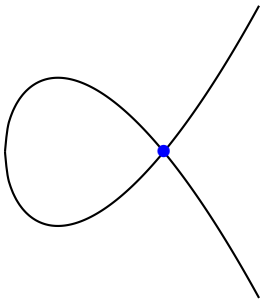


Figure 2.1: Singular curve $y^2 = x^3 - 3x + 2$ over $\mathbb{R}$.

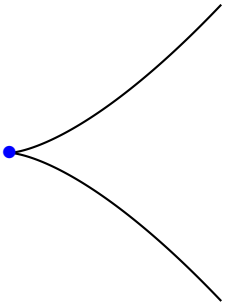


Figure 2.2: Singular curve $y^2 = x^3$ over $\mathbb{R}$.

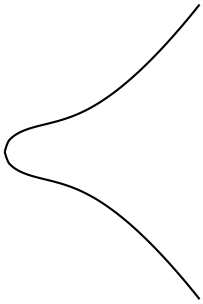


Figure 2.3: Smooth curve $y^2 = x^3 + x + 1$ over $\mathbb{R}$.

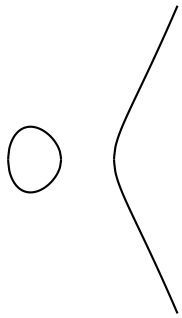


Figure 2.4: Smooth curve $y^2 = x^3 - x$ over $\mathbb{R}$.

2.1.1 The group law: the chord-and-tangent rule

We now turn to describing the elliptic curve group law, and it is here that viewing pictures of elliptic curves over $\mathbb{R}$ is especially instructive. We start with a less formal description until we define the role of the point at infinity $\mathcal{O}$. The group law exploits the fact that, over any field, a line (a degree one equation in x and y) intersects a cubic curve (a degree three equation in x and y) in three places (this is a special case of a more general theorem due to Bezout [Har77, I.7.8]). Namely, if we run a line $\ell : y = \lambda x + \nu$ between two points $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ on E, then substituting this line into $E : y^2 = x^3 + ax + b$ will give a cubic polynomial in x, the roots of which are the x-coordinates of the three points of intersection between ℓ and E. Knowing the two roots (x_P and x_Q) allows us to determine a unique third root that corresponds to the third and only other point in the affine intersection $\ell \cap E$, which we denote by $\ominus R$ (the reason will become clear in a moment). The point $\ominus R$ is then "flipped" over the

x-axis to the point R. In general, the elliptic curve composition law $\oplus$ is defined by this process, namely $R = P \oplus Q$. When computing $R = P \oplus P$, the line ℓ is computed as the tangent to E at P. That is, the derivatives of ℓ and E are matched at P, so (counting multiplicities) ℓ intersects E "twice" at P. Figures 2.5 and 2.6 illustrate why this process is aptly named the *chord-and-tangent* rule.

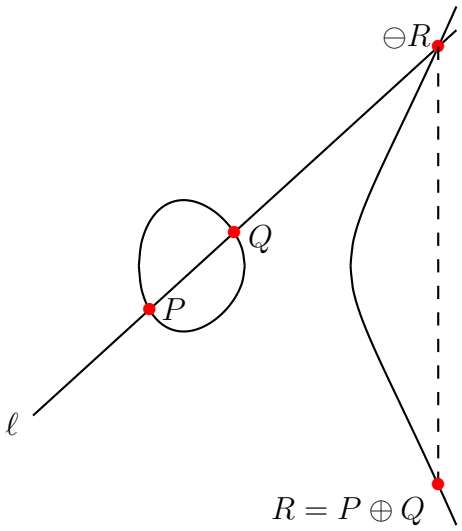


Figure 2.5: Elliptic curve addition.

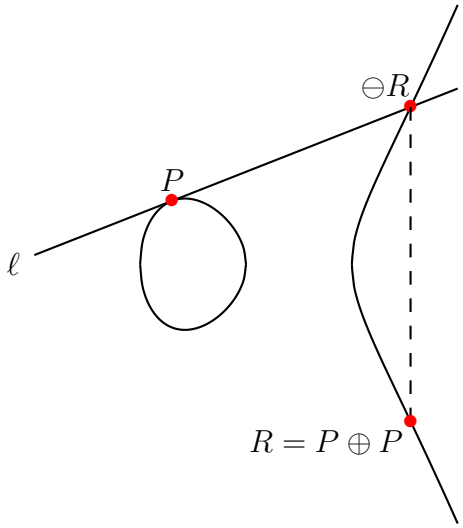


Figure 2.6: Elliptic curve doubling.

Having loosely defined the general group operation, we can now (also loosely) define the role of the point at infinity $\mathcal{O}$. To try and place it somewhere in the above diagrams, one can think of $\mathcal{O}$ as being a point that simultaneously sits infinitely high and infinitely low in the y direction. This allows us to informally conceptualise two properties of elliptic curve groups: firstly, that the point at infinity $\mathcal{O}$ plays the role of the *identity* of the group; and secondly, that the unique inverse of a point is its reflected image over the x-axis (e.g. the $\ominus R$'s in Figures 2.5 and 2.6 are the respective inverses of the R's, and vice versa). If we apply the process in the previous paragraph to compute $R \oplus (\ominus R)$, we start by finding the vertical line that connects them (the dashed lines in Figures 2.5 and 2.6). This line also intersects E (twice) at the point at infinity $\mathcal{O}$, which is then reflected back onto itself, giving $R \oplus (\ominus R) = \mathcal{O}$. Thus, if we define the identity of the group to be $\mathcal{O}$, then the inverse of any element $R = (x_R, y_R)$ is taken as $\ominus R = (x_R, -y_R)$.

Example 2.1.3 (Magma script). $E/\mathbb{R} : y^2 = x^3 - 2x$ is an elliptic curve. The points $(-1, -1)$, $(0, 0)$ and $(2, 2)$ are all on E, and are also on the line $\ell : y = x$.

Applying the technique described above to compute some example group law operations via the line ℓ, we have $(-1,-1) \oplus (0,0) = (2,-2)$, $(2,2) \oplus (0,0) = (-1,1)$, and $(-1,-1) \oplus (2,2) = (0,0)$. All but four points come in pairs with their inverse (i.e. (x',y') and $(x',-y')$); the exceptions being $(0,0)$, $(\sqrt{2},0)$, $(-\sqrt{2},0)$ (notice the vertical tangents when $y=0$ in these cases), and $\mathcal{O}$, which are all their own inverse, e.g. $(0,0) = \ominus(0,0)$, so $(0,0) \oplus (0,0) = \mathcal{O}$ on E. The tangent line ℓ' to E at $(-1,-1)$ is $\ell' : y = -\frac{1}{2}x - \frac{3}{2}$, and it intersects E once more at $(\frac{9}{4}, -\frac{21}{8})$, which gives $(-1,-1) \oplus (-1,-1) = [2](-1,-1) = (\frac{9}{4}, \frac{21}{8})$.

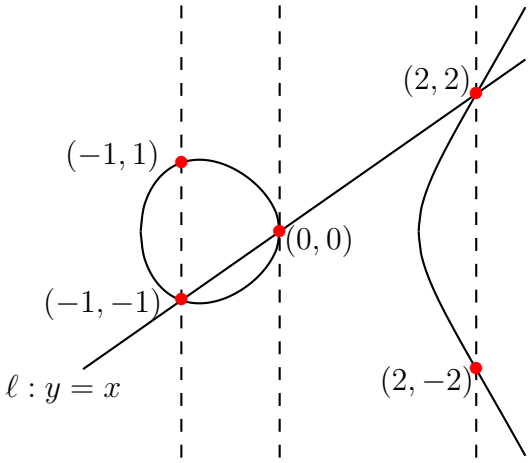


Figure 2.7: Addition in $\mathbb{R}$.

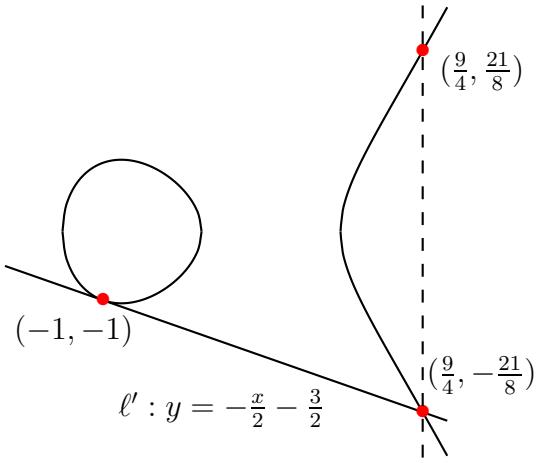


Figure 2.8: Doubling in $\mathbb{R}$.

Example 2.1.4 (Magma script). In this example we consider the same curve equation as the last example, but this time over a small finite field, namely $E/\mathbb{F}_{11} : y^2 = x^3 - 2x$. Rational points are injected naturally across to the finite field case (as long as there is no conflict with the characteristic), so we can immediately find the points $(0,0)$, $(2,2)$ and $(-1,-1) = (10,10)$ (and their inverses) in Figure 2.9. In this case, consider performing the group law operation between the (blue) points $(5,7)$ and $(8,10)$. The line ℓ that joins them is $y = x+2$, which intersects E once more at $(10,1)$. Negating the y-coordinate finds the other point on the dashed line, and gives $(5,7) \oplus (8,10) = (10,10)$.

Example 2.1.4 is also intended to justify why, although (in cryptography) we only ever use elliptic curves over finite fields, we often opt to illustrate the group law by drawing the continuous pictures of curves over $\mathbb{R}$.

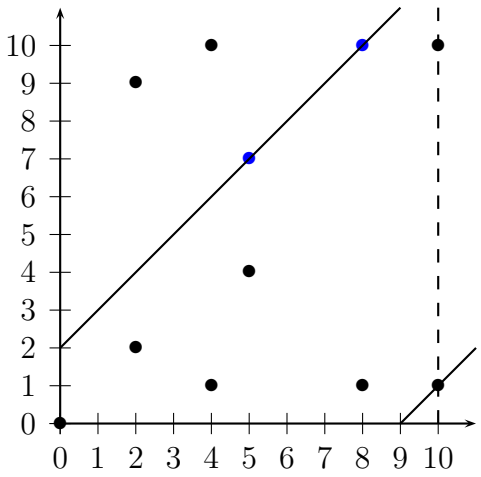
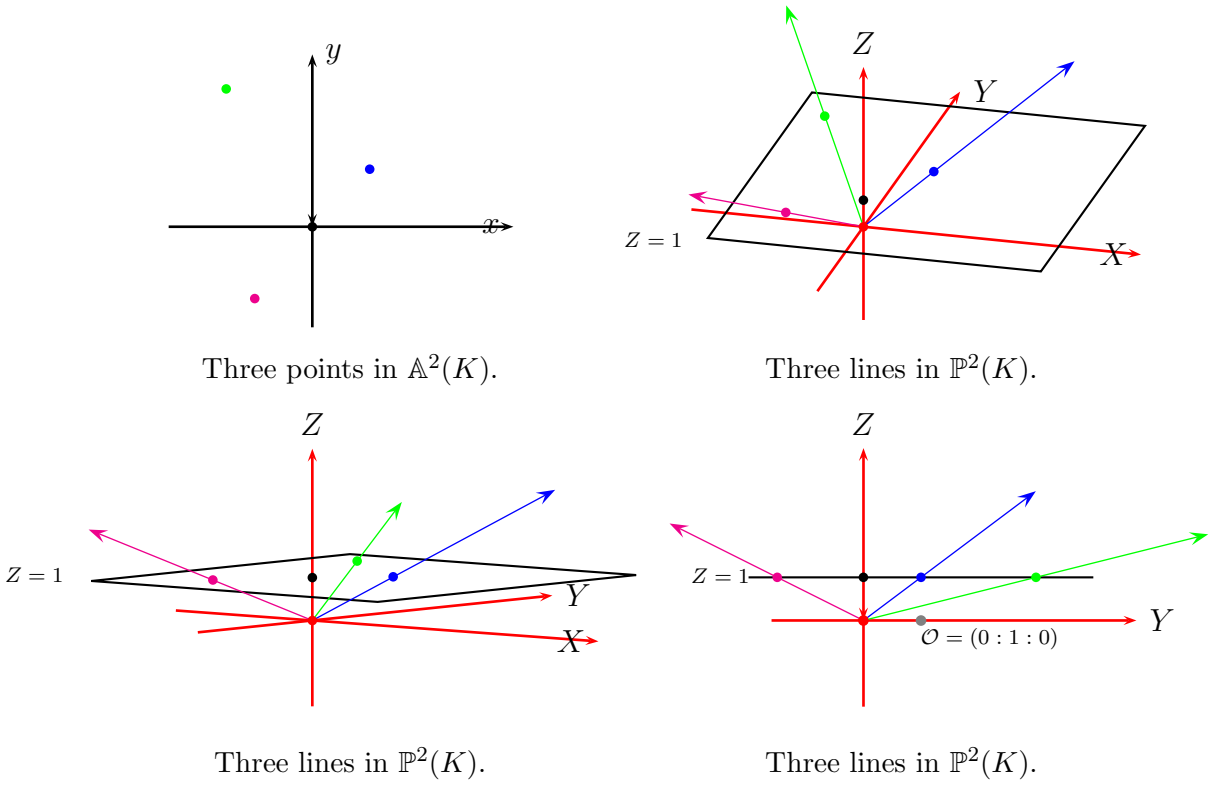


Figure 2.9: The points (excluding $\mathcal{O}$) on $E(\mathbb{F}_{11})$.

The point at infinity in projective space. We now focus our attention on giving a more formal definition for the point at infinity. So far we have been describing elliptic curves in *affine space* as a set of affine points together with the point at infinity: $E = \{(x, y) \in \mathbb{A}^2(\overline{K}) : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$. In general, a more precise way to unify (or include) points at infinity with the affine points is to work in *projective space*: essentially, instead of working with points in n-space, we work with lines that pass through the origin in $(n+1)$-space. For our purposes, this means our affine points in 2-space become lines in 3-space, namely that $(x, y) \in \mathbb{A}^2(\overline{K})$ corresponds to the line defined by all points of the form $(\lambda x, \lambda y, \lambda) \in \mathbb{P}^2(\overline{K})$, where $\lambda \in \overline{K}^*$. That is, $\mathbb{P}^2$ is $\mathbb{A}^3 \setminus \{(0, 0, 0)\}$ modulo the following congruence condition: $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$ if there exists $\lambda \in \overline{K}^*$ such that $(x_1, y_1, z_1) = (\lambda x_2, \lambda y_2, \lambda z_2)$. Figure 2.10 illustrates the relationship between points in $\mathbb{A}^2$ with their congruence classes (lines) in $\mathbb{P}^2$; the lines in 3-space should also extend "downwards" into the region where $Z < 0$ but we omitted this to give more simple pictures. We reiterate that these lines do not include the point $(0, 0, 0)$.

We usually use capital letters and colons to denote a (representative of a) congruence class in projective coordinates, so that in general $(X : Y : Z)$ represents the set of all points on the "line" in $\mathbb{P}^2$ that correspond to $(x, y) \in \mathbb{A}^2$. There are many copies of $\mathbb{A}^2$ in $\mathbb{P}^2$, but we traditionally map the affine point $(x, y) \in \mathbb{A}^2$ to projective space via the trivial inclusion $(x, y) \mapsto (x : y : 1)$, and for any $(X : Y : Z) \neq \mathcal{O} \in \mathbb{P}^2$, we map back to $\mathbb{A}^2$ via $(X : Y : Z) \mapsto (X/Z, Y/Z)$. The point at infinity $\mathcal{O}$ is represented by $(0 : 1 : 0)$ in projective space (see the last



Three points in $\mathbb{A}^2(K)$. Three lines in $\mathbb{P}^2(K)$.

Three lines in $\mathbb{P}^2(K)$. Three lines in $\mathbb{P}^2(K)$.

Figure 2.10: Identifying points in $\mathbb{A}^2$ with lines in $\mathbb{P}^2$

diagram in Figure 2.10), for which we immediately note that the map back to $\mathbb{A}^2$ is ill-defined.

Example 2.1.5 (Magma script). $E/\mathbb{R} : y^2 = x^3 + 3x$ is an elliptic curve. $P = (3, 6) \in \mathbb{A}^2(\overline{\mathbb{R}})$ is a point on E. In projective space, P becomes $P = (3 : 6 : 1) \in \mathbb{P}^2(\overline{\mathbb{R}})$, which represents all points in $(3\lambda, 6\lambda, \lambda)$ for $\lambda \in \overline{\mathbb{R}} \setminus \{0\}$. For example, the points $(12, 24, 4)$, $(-3\sqrt{-1}, -6\sqrt{-1}, -1\sqrt{-1})$, $(3\sqrt{2}, 6\sqrt{2}, \sqrt{2})$ in $\mathbb{A}^3(\overline{\mathbb{R}})$ are all equivalent (modulo the congruence condition) in $\mathbb{P}^2(\overline{\mathbb{R}})$, where they are represented by P. As usual, the point at infinity on E is $\mathcal{O} = (0 : 1 : 0)$.

The way we define the collection of points in projective space is to *homogenise* $E : y^2 = x^3 + ax + b$ by making the substitution $x = X/Z$ and $y = Y/Z$, and multiplying by Z^3 to clear the denominators, which gives

$$E_{\mathbb{P}} : Y^2Z = X^3 + aXZ^2 + bZ^3. \tag{2.3}$$

The set of points (X, Y, Z) with coordinates in $\overline{K}$ that satisfies (2.3) is called the *projective closure* of E. Notice that $(0, \lambda, 0)$ is in the projective closure for all $\lambda \in \overline{K}^*$, and that all such points cannot be mapped into $\mathbb{A}^2$, justifying the

representative of point at infinity being $\mathcal{O} = (0 : 1 : 0)$.

Example 2.1.6 (Magma script). Consider $E/\mathbb{F}_{13} : y^2 = x^3 + 5$. There are 15 affine points $(x, y) \in \mathbb{A}^2(\mathbb{F}_{13})$ on E, which (with the point at infinity $\mathcal{O}$) gives $\#E(\mathbb{F}_{13}) = 16$. On the other hand, if we homogenise (or projectify) E to give $E_{\mathbb{P}}/\mathbb{F}_{13} : Y^2Z = X^3 + 5Z^3$, then there are 16 classes $(X : Y : Z) \in \mathbb{P}^2(\mathbb{F}_{13})$: $(0 : 1 : 0)$, $(2 : 0 : 1)$, $(4 : 2 : 1)$, $(4 : 11 : 1)$, $(5 : 0 : 1)$, $(6 : 0 : 1)$, $(7 : 6 : 1)$, $(7 : 7 : 1)$, $(8 : 6 : 1)$, $(8 : 7 : 1)$, $(10 : 2 : 1)$, $(10 : 11 : 1)$, $(11 : 6 : 1)$, $(11 : 7 : 1)$, $(12 : 2 : 1)$, $(12 : 11 : 1)$. Each of these classes represents several points $(X, Y, Z) \in \mathbb{A}^3(\mathbb{F}_{13})$ whose coordinates satisfy $Y^2Z = X^3 + 5Z^3$ (there are actually 195 such points, but this is not important). In fact, each class represents infinitely many points on $E_{\mathbb{P}}(\overline{\mathbb{F}}_{13})$. Any reader that is familiar with Magma, or has been working through our examples with the accompanying Magma scripts, will recognise the representation of points as representatives in $\mathbb{P}^2$.

The projective coordinates (X, Y, Z) used to replace the affine coordinates (x, y) above are called *homogenous projective coordinates*, because the projective version of the curve equation in (2.3) is homogeneous. These substitutions ($x = X/Z$, $y = Y/Z$) are the most simple (and standard) way to obtain projective coordinates, but we are not restricted to this choice of substitution. In Chapter 3 of this thesis, we will be exploring several different projective options, but the recipe for projectifying remains the same. Namely, we will always be using (the most natural) projections obtained through substitutions of the form $x = X/Z^i$ and $y = Y/Z^j$.

Example 2.1.7 (Magma script). Consider $E/\mathbb{F}_{41} : y^2 = x^3 + 4x - 1$. Using homogeneous coordinates gives rise to the projective equation $Y^2Z = X^3 + 4XZ^2 - Z^3$, with the point at infinity being $\mathcal{O} = (0 : 1 : 0)$. Alternatively, a projection we make use of in this thesis is $x = X/Z$ and $y = Y/Z^2$, which in this instance give the projective equation $Y^2 = X^3Z + 4XZ^3 - Z^4$, from which the point at infinity is seen (from putting $Z = 0$) to be $\mathcal{O} = (1 : 0 : 0)$. Another commonly used coordinate system is Jacobian coordinates, which use the substitutions $x = X/Z^2$ and $y = Y/Z^3$ to give the projective equation $Y^2 = X^3 + 4XZ^4 - Z^6$. In this case, we substitute $Z = 0$ to see that the point at infinity is defined by the line $\mathcal{O} = (\lambda^2 : \lambda^3 : 0) \in \mathbb{P}^2(\mathbb{F}_{41})$.

Deriving (affine) explicit formulas for group law computations. We are now in a position to give explicit formulas for computing the elliptic curve

group law. The chord-and-tangent process that is summarised in Figures 2.5 and 2.6 allows a simple derivation of these formulas. We derive the formulas in affine space, but will soon transfer them into projective space as well. The derivation of the formulas for point additions $R = P \oplus Q$ and for point doublings $R = P \oplus P$ follow the same recipe, the main difference being in the calculation of the gradient λ of the line $\ell : y = \lambda x + \nu$ that is used. We will first derive the formulas for the addition $R = P \oplus Q$ in the general case, and will then make appropriate changes for the general doubling formulas. By "general case", we mean group law operations between points where neither point is $\mathcal{O}$, and the points that are being added are not each inverses of one another; we will handle these special cases immediately after the general cases. Referring back to Figure 2.5, the line $\ell : y = \lambda x + \nu$ that intersects $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ has gradient $\lambda = (y_Q - y_P)/(x_Q - x_P)$. From here, ν can simply be calculated as either $\nu = y_P - \lambda x_P$ or $\nu = y_Q - \lambda x_Q$, but in the literature we will often see an unbiased average of the two as $\nu = (y_Q x_P - y_P x_Q)/(x_P - x_Q)$. From here we substitute $\ell : y = \lambda x + \nu$ into $E : y^2 = x^3 + ax + b$ to find the third affine point of intersection, $\ominus R$, in $\ell \cap E$. Finding the coordinates of $\ominus R$ trivially reveals the coordinates of $R = (x_R, y_R)$, since $\ominus R = (x_R, -y_R)$; the roots of the cubic that result will be x_P, x_Q and x_R. Namely,

$$\begin{aligned}(x - x_P)(x - x_Q)(x - x_R) &= (x^3 + ax + b) - (\lambda x + \nu)^2 \\ &= x^3 - \lambda^2 x^2 + (a - 2\lambda\nu)x + b - \nu^2.\end{aligned}$$

We only need to look at the coefficient of x^2 to determine x_R, since the coefficient on the left hand side is $-(x_P + x_Q + x_R)$. From here, recovering the y-coordinate is simple, since $-y_R$ lies on ℓ, so

$$x_R = \lambda^2 - x_P - x_Q; \qquad\qquad y_R = -(\lambda x_R + \nu).$$

This finishes the description of addition in the general case. When adding P to itself (i.e. doubling P – refer back to Figure 2.6), the line $\ell : y = \lambda x + \nu$ is the tangent to E at P. Thus, its gradient λ is the derivative function dy/dx of E, evaluated at P. To obtain dy/dx, we differentiate the curve equation implicitly,

as

$$\begin{aligned}\frac{d}{dx}(y^2) &= \frac{d}{dx}(x^3 + ax + b)\\ \frac{d}{dy}(y^2)\frac{dy}{dx} &= 3x^2 + a\\ \frac{dy}{dx} &= \frac{3x^2 + a}{2y}.\end{aligned}$$

Thus, $\lambda = \frac{dy}{dx}(P) = (3x_P^2 + a)/(2y_P)$, and $\nu = y_P - \lambda x_P$. Again, we substitute ℓ into E, but this time two of the roots of the resulting cubic are x_P, so we obtain x_R and y_R as

$$x_R = \lambda^2 - 2x_P; \qquad\qquad y_R = -(\lambda x_R + \nu).$$

This finishes the derivation of doubling formulas in the general case. We now complete the group law description by looking at the special cases. The point at infinity $\mathcal{O}$ is the identity, or neutral element, so any operation involving it is trivial. Otherwise, any operation between elements P and Q with different x-coordinates employs the general addition. This leaves the remaining cases of $x_P = x_Q$: (i) if $y_P = -y_Q$, then P and Q are inverses of each other and $P \oplus Q = \mathcal{O}$ (note that this includes $y_P = y_Q = 0$), and (ii) if $y_P = y_Q \neq 0$, then $P = Q$ and we use the point doubling formulas.

Much of the literature concerning the elliptic curve group law tends to present the complete description in the previous paragraph using an "if-then-else" style algorithm, where the "if" statements distinguish which of the above scenarios we are in. In optimised cryptographic implementations however, this is not the way that the group law operation is coded. This is because the groups we use are so large that the chances of running into a special case (that is not general doubling or general addition) randomly is negligible. Moreover, the parameters are usually chosen so that we are guaranteed not to run into these cases. In this light then, it will soon become clear that the major operations we are concerned with are point additions $R = P \oplus Q$ and point doublings $R = P \oplus P$, the formulas for which are summarised in (2.4) and (2.5) respectively.

(Affine addition) $$\lambda = \frac{y_Q - y_P}{x_Q - x_P}; \qquad \nu = y_P - \lambda x_P;$$

$$(x_P, y_P) \oplus (x_Q, y_Q) = (x_R, y_R) = \left(\lambda^2 - x_P - x_Q, -(\lambda x_R + \nu)\right). \quad (2.4)$$

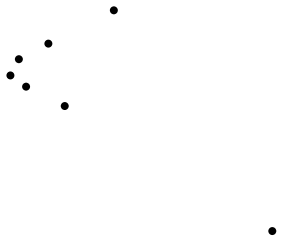
$$\text{(Affine doubling)} \qquad \lambda = \frac{3x_P^2 + a}{2y_P}; \qquad \nu = y_P - \lambda x_P;$$

$$[2](x_P, y_P) = (x_P, y_P) \oplus (x_P, y_P) = (x_R, y_R) = \left(\lambda^2 - 2x_P, -(\lambda x_R + \nu)\right). \quad (2.5)$$

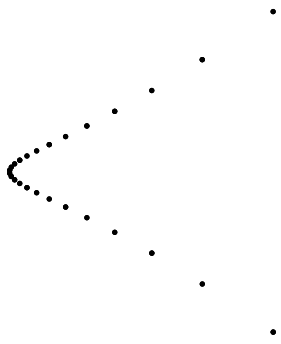
Example 2.1.8 (Magma script). We revisit the curve $E/\mathbb{Q} : y^2 = x^3 - 2$ from Example 2.1.1 to verify the group law calculations that were stated. We start with the point doubling of $P = (x_P, y_P) = (3, 5)$, to compute $Q = [2]P = P \oplus P$ using (2.5). Here, $\lambda = \frac{3x_P^2+a}{2y_P} = \frac{3\cdot 3^2+0}{2\cdot 5} = \frac{27}{10}$, from which ν follows as $\nu = y_P - \lambda x_P = 5 - \frac{27}{10} \cdot 3 = -\frac{31}{10}$. Thus, $x_Q = \lambda^2 - 2x_P = (\frac{27}{10})^2 - 2 \cdot 3 = \frac{129}{100}$, and $y_Q = -(\lambda x_Q + \nu) = -(\frac{27}{10} \cdot \frac{129}{100} - \frac{31}{10}) = -\frac{383}{1000}$, giving $(x_Q, y_Q) = [2](x_P, y_P) = (\frac{129}{100}, -\frac{383}{1000})$. For the addition $R = P \oplus Q$, we use the formulas in (2.4), so $\lambda = \frac{y_Q - y_P}{x_Q - x_P} = (-\frac{383}{1000} - 5)/(\frac{129}{100} - 3) = \frac{5383}{1710}$, and $\nu = y_P - \lambda x_P = 5 - \frac{5383}{1710} \cdot 3 = -\frac{2533}{570}$. Thus, $x_R = \lambda^2 - x_P - x_Q = (\frac{5383}{1710})^2 - 3 - \frac{129}{100} = \frac{164323}{29241}$, and $y_R = \lambda x_R + \nu = \frac{5383}{1710} \cdot \frac{164323}{29241} - \frac{2533}{570} = -\frac{66234835}{5000211}$, so $(x_R, y_R) = (\frac{164323}{29241}, -\frac{66234835}{5000211})$. Since $Q = [2]P = P \oplus P$, then $R = P \oplus Q = [3]P$. We finish this example with a remark that further justifies the use of finite fields as the underlying fields in cryptography. It is not too painful to show that $P = (3, 5)$ and $\ominus P = (3, -5)$ are the only integral points on E [Sil09, Ch. IX, Prop. 7.1(b)], or that $E(\mathbb{Q})$ is actually *infinite cyclic* [Sil09, Ch. IX, Remark 7.1.1], meaning that among infinitely many rational points, only two have integer coordinates. Besides the infinite nature of $E(\mathbb{Q})$ (the lack of any finite subgroups is not useful in the context of discrete logarithm based cryptographic groups), observing the growing size of the numerators and denominators in $[n]P$, even for very small values of n, shows why using $E(\mathbb{Q})$ would be impractical. Using Magma, we can see that the denominator of the y-coordinate of $[10]P$ is 290 bits, whilst the denominator in $[100]P$ is 29201 bits, which agrees with the group law formulas in (2.4) and (2.5) that suggest that denominators of successive scalar multiples of P would grow quadratically; even Magma takes its time computing $[1000]P$, whose denominator is 2920540 bits, and Magma could not handle the computation of $[10000]P$. In Figure 2.11 we plot multiples of $P = (3, 5)$ that fall within the domain $x < 6$.

From now on we will only be working with elliptic curves over finite fields. We start with a simple example of basic group law computations on $E(\mathbb{F}_q)$ to summarise the discussion up until this point.

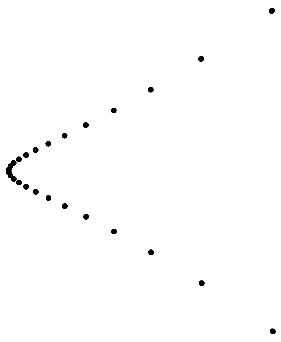
Example 2.1.9 (Magma script). $E/\mathbb{F}_{23} : y^2 = x^3 + 5x + 7$ is an elliptic curve, and



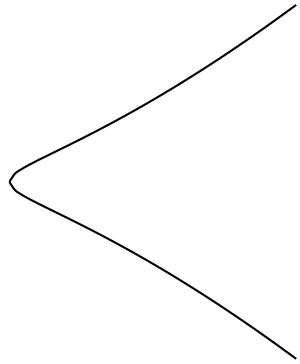
Of the first 10 multiples of $P = (3, 5)$ in $E(\mathbb{Q})$, 7 had $x < 6$.



Of the first 100 multiples of $P = (3, 5)$ in $E(\mathbb{Q})$, 64 had $x < 6$.



Of the first 1000 multiples of $P = (3, 5)$ in $E(\mathbb{Q})$, 635 had $x < 6$.



$E : y^2 = x^3 - 2$ over $\mathbb{R}$.

Figure 2.11: More and more points (with $x < 6$) in the infinite group $E(\mathbb{Q})$

both $P = (x_P, y_P) = (2, 5)$ and $Q = (x_Q, y_Q) = (12, 1)$ are on E. Using the affine point addition formulas in (2.4), we find $R = P \oplus Q$ by first computing $\lambda = \frac{y_Q - y_P}{x_Q - x_P} = \frac{1-5}{12-2} = -4 \cdot 10^{-1} = -28 = 18$, from which ν follows as $\nu = y_P - \lambda x_P = 5 - 18 \cdot 2 = -31 = 15$, so $\ell : y = 18x + 15$ is the line running through P and Q. We then compute $(x_R, y_R) = (\lambda^2 - x_P - x_Q, -(\lambda x_R + \nu))$, so $x_R = 18^2 - 2 - 12 = 11$ and $y_R = -(18 \cdot 11 + 15) = 17$, meaning $R = (11, 17)$. Applying (2.5) to compute $S = [2]P$ gives $\lambda' = \frac{3x_P^2 + a}{2y_P} = \frac{3 \cdot 2^2 + 5}{2 \cdot 5} = 17 \cdot 10^{-1} = 17 \cdot 7 = 4$, and ν' follows as $\nu' = y_P - \lambda' x_P = 5 - 4 \cdot 2 = 20$, so $\ell' : y = 4x + 20$ is the tangent line that intersects E with multiplicity two at P. We then compute $(x_S, y_S) = (\lambda'^2 - 2x_P, -(\lambda' x_S + \nu'))$, so $x_S = 4^2 - 2 \cdot 2 = 12$ and $y_S = -(4 \cdot 12 + 20) = -68 = 1$, meaning $S = (12, 1)$.

We now give an example of the *multiplication-by-m* map on E, defined as

$$[m] : E \to E, \qquad P \mapsto [m]P,$$

and illustrate the straightforward way to compute it in practice. This operation is analogous to exponentiation $g \mapsto g^m$ in $\mathbb{Z}_q^*$, and is the central operation in ECC, as it is the *one-way* operation that buries discrete logarithm problems in $E(\mathbb{F}_q)$. To efficiently compute the exponentiation g^m in $\mathbb{Z}_q^*$, we *square-and-multiply*, whilst to compute the scalar multiplication $[m]P$ in $E(\mathbb{F}_q)$, we (because of the additive notation) *double-and-add*.

Example 2.1.10 (Magma script). Let $E/\mathbb{F}_{1021} : y^2 = x^3 - 3x - 3$ so that $r = \#E(\mathbb{F}_q) = 1039$ is prime. Let $P = (379, 1011) \in E$ and $m = 655$, and suppose we are to compute $[m]P = [655](379, 1011)$. To double-and-add, we write the (10-bit) binary representation of m as $m = (m_9, ..., m_0)_2 = (1, 0, 1, 0, 0, 0, 1, 1, 1, 1)$. Initialising $T \leftarrow P$, and starting from the second most significant bit m_8, we successively compute $T \leftarrow [2]T$ for each bit down to m_0, and whenever $m_i = 1$ we compute $T \leftarrow T+P$. So, in our case it takes 9 doublings $T \leftarrow [2]T$ and 5 additions $T \leftarrow T + P$ to compute $[m]P$, which ends up being $[655](379, 1011) = (388, 60)$. In general then, this straightforward double-and-add algorithm will take $\log_2 m$ doublings and roughly half as many additions to compute $[m]P$ (if m is randomly chosen).

The group axioms. All but one of the group axioms are now concrete. Namely, for *closure*, if we start with two points in $E(K)$, then the chord-and-tangent process gives rise to a cubic polynomial in K for which two roots (the two x-coordinates of the points we started with) are in K, meaning the third root must also be in K; the explicit formulas affirm this. The *identity* and *inverse* axioms are fine, since $P \oplus \mathcal{O} = P$, and the element $\ominus P$ such that $P \oplus (\ominus P) = \mathcal{O}$ is clearly unique and well defined for all P. We also note that the group is *abelian*, since the process of computing $P \oplus Q$ is symmetric. The only non-obvious axiom is *associativity*, i.e. showing $(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$. An elementary approach using the explicit formulas above can be used to show associativity by treating all the separate cases, but this approach is rather messy [Fri05]. Silverman gives a much more instructive proof [Sil09, Ch. III.3.4e] using tools that we will develop in the following section, but for now we offer some temporary intuition via the illustration in Figures 2.12 and 2.13.

Speeding up elliptic curve computations. Group law computations on elliptic curves are clearly more complicated than computations in traditional

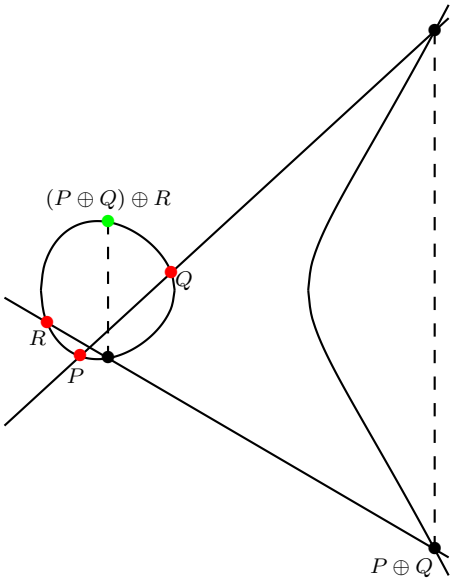


Figure 2.12: $(P \oplus Q) \oplus R$.

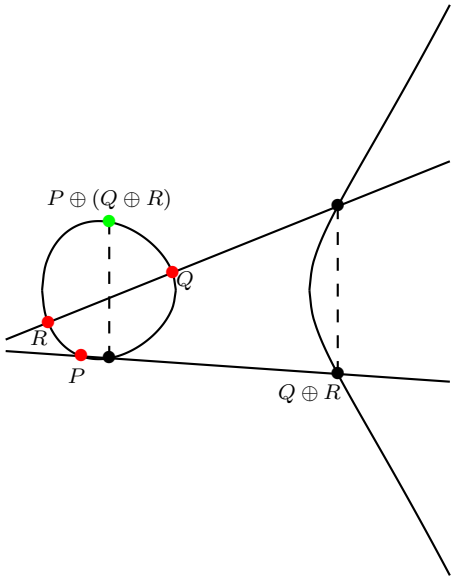


Figure 2.13: $P \oplus (Q \oplus R)$.

groups that facilitate discrete logarithm based protocols like $\mathbb{F}_q^*$; the explicit formulas in (2.4) and (2.5) use many field operations. However, in the context of cryptography, the more abstract nature of elliptic curve groups actually works in their favour. This is essentially because attackers aiming to solve the discrete logarithm problem on elliptic curves also face this abstractness. The subexponential algorithms that apply to finite field discrete logarithms[2] do not translate to the elliptic curve setting, where the best available attacks remain generic, exponential algorithms like Pollard rho [Pol78]. This means that elliptic curve groups of a relatively small size achieves the same conjectured security as multiplicative groups in much larger finite fields, i.e. $E(\mathbb{F}_{q_1})$ and $\mathbb{F}_{q_2}^*$ achieve similar security when $q_2 \gg q_1$. For example, an elliptic curve defined over a 160-bit field currently offers security comparable to a finite field of 1248 bits [Sma10, Table 7.2]. Thus, although more field operations are required to perform a group law computation, these operations take place in a field whose operational complexity is much less, and this difference is more than enough to tip the balance in the favour of elliptic curves. In addition, the smaller group elements in $E(\mathbb{F}_{q_1})$ implies much smaller key sizes, greatly reducing storage and bandwidth requirements. These are some of the major reasons that elliptic curves have received so much attention in the realm of public-key cryptography; the field of elliptic curve cryptography (ECC) has been thriving since Koblitz [Kob87] and Miller [Mil85]

[2]See Diem's notes on *index calculus* for a nice introduction [Die12].

independently suggested their potential as alternatives to traditional groups.

One avenue of research that has given ECC a great boost is that of optimising the group law computations. The explicit formulas in affine coordinates ((2.4) and (2.5)) would not be used to compute the group law in practice, and in fact the Weierstrass model $E : y^2 = x^3 + ax + b$ is often not the optimal curve model either. A huge amount of effort has been put towards investigating other models and coordinate systems in order to minimise the field operations required in group law computations. One of the initial leaps forward in this line of research was the observation that performing computations in projective space avoids field inversions, which are extremely costly in practice. We illustrate these techniques in the following examples.

Example 2.1.11 (Magma script). Consider a general Weierstrass curve $E(\mathbb{F}_q) : y^2 = x^3 + ax + b$ where q is a large prime, and let $\mathbf{M}$, $\mathbf{S}$ and $\mathbf{I}$ represent the cost of computing multiplications, squarings and inversions in $\mathbb{F}_q$ respectively. To compute a general affine point doubling $(x_R, y_R) = [2](x_P, y_P)$ using (2.5) costs $2\mathbf{M}+2\mathbf{S}+\mathbf{I}$, and to compute a general affine point addition $(x_R, y_R) = (x_P, y_P) \oplus (x_Q, y_Q)$ using (2.4) costs $2\mathbf{M} + \mathbf{S} + \mathbf{I}$. On the other hand, we can transform the formulas into homogeneous projective space according to the substitutions $x = X/Z$ and $y = Y/Z$, and we can consider computing $(X_R : Y_R : Z_R) = [2](X_P : Y_P : Z_P)$ and $(X_R : Y_R : Z_R) = (X_P : Y_P : Z_P) \oplus (X_Q : Y_Q : Z_Q)$ on $E : Y^2Z = X^3 + aXZ^2 + bZ^3$. For the addition case, substituting $x_i = X_i/Z_i$ and $y_i = Y_i/Z_i$ for $i \in \{P, Q, R\}$ into the affine formulas

$$x_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)^2 - x_P - x_Q; \qquad y_R = \left(\frac{y_Q - y_P}{x_Q - x_P}\right)(x_P - x_R) - y_P$$

taken from (2.4), gives

$$\frac{X_R}{Z_R} = \left(\frac{\frac{Y_Q}{Z_Q} - \frac{Y_P}{Z_P}}{\frac{X_Q}{Z_Q} - \frac{X_P}{Z_P}}\right)^2 - \frac{X_P}{Z_P} - \frac{X_Q}{Z_Q}; \qquad \frac{Y_R}{Z_R} = \left(\frac{\frac{Y_Q}{Z_Q} - \frac{Y_P}{Z_P}}{\frac{X_Q}{Z_Q} - \frac{X_P}{Z_P}}\right)\left(\frac{X_P}{Z_P} - \frac{X_R}{Z_R}\right) - \frac{Y_P}{Z_P}.$$

After a little manipulation, we can then set Z_R to be the smallest value that contains both denominators above, and update the numerators accordingly to

give

$$
\begin{aligned}
X_R &= (X_P Z_Q - X_Q Z_P)\left(Z_P Z_Q (Y_P Z_Q - Y_Q Z_P)^2 - (X_P Z_Q - X_Q Z_P)^2 (X_P Z_Q + X_Q Z_P)\right); \\
Y_R &= Z_P Z_Q (X_Q Y_P - X_P Y_Q)(X_P Z_Q - X_Q Z_P)^2 \\
&\quad - (Y_P Z_Q - Y_Q Z_P)\left((Y_P Z_Q - Y_Q Z_P)^2 Z_P Z_Q - (X_P Z_Q + X_Q Z_P)(X_P Z_Q - X_Q Z_P)^2\right); \\
Z_R &= Z_P Z_Q (X_P Z_Q - X_Q Z_P)^3.
\end{aligned}
$$

The explicit formulas database (EFD) [BL07a] reports that the above formulas can be computed in a total of $12\mathbf{M} + 2\mathbf{S}$. The real power of adopting projective coordinates for computations becomes apparent when we remark that most optimised implementations of $\mathbb{F}_q$ arithmetic have $\mathbf{I} \gg 20\mathbf{M}$, and the multiplication to inversion ratio is commonly reported to be $80:1$ or higher. Thus, the $12\mathbf{M} + 2\mathbf{S}$ used for additions in projective space will be much faster than the $2\mathbf{M} + \mathbf{S} + \mathbf{I}$ for affine additions. For completeness, we remark that deriving the projective formulas for computing $(X_R : Y_R : Z_R) = [2](X_P : Y_P : Z_P)$ is analogous (but substantially more compact since we only have the projective coordinates of P to deal with), and the EFD reports that this can be done in $5\mathbf{M} + 6\mathbf{S}$, which will again be much faster than the $2\mathbf{M} + 2\mathbf{S} + I$ in affine space.

The Weierstrass model for elliptic curves covers all isomorphism classes, meaning that every elliptic curve can be written in Weierstrass form. Other models of elliptic curves are usually available if some condition holds, and (if this is the case) it can be advantageous to adopt such a model, as the following example shows.

Example 2.1.12 (Magma script). If $x^3 + ax + b$ has a root in $\mathbb{F}_q$, then Billet and Joye [BJ03, Eq. 8-10] show that instead of working with $E : y^2 = x^3 + ax + b$, we can work with the (birationally equivalent) *Jacobi-quartic* curve $J : v^2 = au^4 + du^2 + 1$, for appropriately defined a, d (that depend on the root). Here we write J using (u, v) coordinates so back-and-forth mappings are defined without confusion. Thus, consider $E/\mathbb{F}_{97} : y^2 = x^3 + 5x + 5$, for which $x^3 + 5x + 5$ has 34 as a root, so we will work on the isomorphic curve $J/\mathbb{F}_{97} : v^2 = 73u^4 + 46u^2 + 1$. Instead of homogeneous projective coordinates, [BJ03] projectified under the substitution $u = U/W$ and $v = V/W^2$, which gives the (non-homogeneous) projective closure as $J : V^2 = 73U^4 + 46U^2W^2 + W^4$. Any point $(x, y) \neq \mathcal{O}$ on E can be taken straight to the projective closure of J via

$$
(x, y) \mapsto \left(2(x - 34) : (2x + 34)(x - 34)^2 - y^2 : y\right),
$$

with the reverse mapping given by

$$(U : V : W) \mapsto \left(2\frac{V + W^2}{U^2} - 17, W\frac{4(V + W^2) - 5U^2}{U^3}\right).$$

For example $(x, y) = (77, 21)$ maps to $(U : V : W) = (86 : 8 : 21)$, and vice versa. We now look at the formulas for the point addition $(U_3 : V_3 : W_3) = (U_1 : V_1 : W_1) \oplus (U_2 : V_2 : W_2)$ on $J : V^2 = aU^4 + dU^2W^2 + W^4$, taken from [BJ03, Eq. 11], as

$$\begin{aligned}
U_3 &= U_1W_1V_2 + U_2W_2V_1, \\
V_3 &= \left((W_1W_2)^2 + a(U_1U_2)^2\right)(V_1V_2 + dU_1U_2W_1W_2) + 2aU_1U_2W_1W_2(U_1^2W_2^2 + U_2^2W_1^2), \\
W_3 &= (W_1W_2)^2 - a(U_1U_2)^2,
\end{aligned}$$

where we immediately highlight the relative simplicity of the above formulas in comparison to the homogeneous projective formulas derived in the previous example. Unsurprisingly then, the fastest formulas for Jacobi-quartic additions and doublings outdo those for general Weierstrass curves in homogeneous projective space. Namely, the current fastest formulas for doublings on Jacobi-quartics cost $2\mathbf{M} + 5\mathbf{S}$ and additions cost $6\mathbf{M} + 4\mathbf{S}$ [HWCD09], whilst in the previous example we had $5\mathbf{M} + 6\mathbf{S}$ for doublings and $12\mathbf{M} + 2\mathbf{S}$ for additions.

The Jacobi-quartic curves discussed above are just one example of dozens of models that have been successful in achieving fast group law computations, and therefore fast cryptographic implementations. Other well known models include Edwards curves [Edw07, BL07b], Hessian curves [JQ01, Sma01] and Montgomery curves [Mon87]. We refer to the EFD [BL07a] for a catalogue of all the fastest formulas for the popular curve models, and to Hisil's thesis [His10] for a general method of (automatically) deriving fast group law algorithms on arbitrary curve models. For any reader wishing to delve even further into group law arithmetic on elliptic curves, we also recommend the recent, advanced works by Castryck and Vercauteren [CV11], and by Kohel [Koh11].

2.1.2 Torsion, endomorphisms and point counting

We now turn our focus to the behaviour of elliptic curve groups, as they are used in cryptography. We start by importantly discussing the possible structures exhibited by the finite group $E(\mathbb{F}_q)$. It turns out that $E(\mathbb{F}_q)$ is either itself cyclic,

or isomorphic to a product of two cyclic groups $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ with $n_1 \mid n_2$ [ACD+05, Prop. 5.78]. In cryptography, we would like the group $E(\mathbb{F}_q)$ to be *as cyclic as possible*, so we usually prefer the former case, or at the very least for n_1 to be very small. In most cases of practical interest, we can generate curves that are cyclic with relative ease, so throughout this thesis it is to safe assume that $E(\mathbb{F}_q)$ is cyclic (but to see the real depth of this question in general, we refer to [MS07]). The following example illustrates that $E(\mathbb{F}_q) = \langle P \rangle$ obeys all the usual rules that apply to cyclic groups, and introduces the important notion of *r-torsion*.

Example 2.1.13 (Magma script). Consider $E/\mathbb{F}_{101} : y^2 = x^3 + x + 1$. The group order is $\#E(\mathbb{F}_q) = 105 = 3 \cdot 5 \cdot 7$, and $P = (47, 12) \in E$ is a generator. Lagrange's theorem says that points (and subgroups) over the base field will have order in $\{1, 3, 5, 7, 15, 21, 35, 105\}$. Indeed, to get a point of order $r \mid 105$, we simply multiply P by the appropriate *cofactor*, which is $h = \#E/r$. For example, a point of order 3 is $[35](47, 12) = (28, 8)$, a point of order 21 is $[5](47, 12) = (55, 65)$, and a point of order 1 is $[105](47, 12) = \mathcal{O}$ (which is the only such point). By definition, a point is "killed" (sent to $\mathcal{O}$) when multiplied by its order. Any point over the full closure $E(\overline{\mathbb{F}}_q)$ that is killed by r is said to be in the r-torsion. So, the point $(55, 65)$ above is in the 21-torsion, as is the point $(28, 8)$. There are exactly 21 points in $E(\mathbb{F}_q)$ in the 21-torsion, but there are many more in $E(\overline{\mathbb{F}}_q)$.

The whereabouts and structure of r-torsion points in $E(\overline{\mathbb{F}}_q)$ (alluded to at the end of Example 2.1.13) plays a crucial role in pairing-based cryptography; we will be looking at this in close detail in Section 2.3.

In ECC we would like the group order $\#E(\mathbb{F}_q)$ to be as close to prime as possible. This is because the (asymptotic) complexity of the ECDLP that attackers face is dependent on the size of the largest prime subgroup of $E(\mathbb{F}_q)$. Even if the particular instance of the discrete logarithm problem uses a generator of the whole group, the attacker can use the known group order to solve smaller instances in subgroups whose orders are pairwise prime, and then reconstruct the answer using the Chinese Remainder Theorem (CRT). We make this clear in the following two examples: the first is a toy example, whilst the second shows the difference between two curves of the same cryptographic size; one that is currently considered secure and one that is completely breakable using modern attacks.

Example 2.1.14 (Magma script). Consider $E/\mathbb{F}_{1021} : y^2 = x^3 + 905x + 100$, with

group order $\#E(\mathbb{F}_q) = 966 = 2 \cdot 3 \cdot 7 \cdot 23$, and generator $P = (1006, 416)$. Suppose we are presented with an instance of the ECDLP: namely, we are given $Q = (612, 827)$, and we seek to find k such that $[k]P = Q$. For the sake of the example, suppose our best "attack" is trivial: trying every multiple $[i]P$ of P until we hit the correct one ($i = k$). Rather than seeking i in the full group ($2 \leq i \leq 965$), we can map the instance into each prime order subgroup by multiplying by the appropriate cofactor, and then solve for $k_j \equiv k \bmod j$, $j \in \{2, 3, 7, 23\}$. For $j = 2$, we have $P_j = P_2 = [966/2]P = [483](1006, 416) = (174, 0)$, and $Q_j = Q_2 = [483](612, 827) = (174, 0)$, so $Q_2 = [k_2]P_2$ gives $k_2 = 1$. For $j = 3$, we have $P_3 = [322]P = (147, 933)$ and $Q_3 = [322]P = \mathcal{O}$, so $Q_3 = [k_3]P_3$ gives $k_3 = 3$. For $j = 7$, we have $P_7 = [138]P = (906, 201)$ and $Q_7 = [138]Q = (906, 201)$, so $Q_7 = [k_7]P_7$ gives $k_7 = 1$. For $j = 23$, we have $P_{23} = [42]P = (890, 665)$ and $Q_{23} = [42]Q = (68, 281)$. For $Q_{23} = [k_{23}]P_{23}$, we exhaust $k_{23} \in \{1, .., 22\}$ to see that $k_{23} = 20$. Now, we can use the Chinese Remainder Theorem to solve

$$k \equiv k_2 = 1 \bmod 2; \quad k \equiv k_3 = 0 \bmod 3; \quad k \equiv k_7 = 1 \bmod 7; \quad k \equiv k_{23} = 20 \bmod 23,$$

which gives $k \equiv 687 \bmod \#E$, solving the ECDLP instance. Notice that the hardest part was exhausting the set $\{1, .., 22\}$ to find $k_{23} = 20$, so the largest prime order subgroup becomes the bottleneck of the algorithm, giving intuition as to why the largest prime order subgroup defines the attack complexity when groups of a cryptographic size are used.

Example 2.1.15 (Magma script). For our real world example, we take the curve P-256 from the NIST recommendations [NIS99], which currently achieves a similar security level (resistance against best known attacks) to the 128-bit Advanced Encryption Standard (AES) for symmetric encryption. The curve is defined as $E/\mathbb{F}_q : y^2 = x^3 - 3x + b$, with prime order $r = \#E$, and generator $G = (x_G, y_G)$, where

$q = 115792089210356248762697446949407573530086143415290314195533631308867097853951,$
$r = 115792089210356248762697446949407573529996955224135760342422259061068512044369,$
$b = 41058363725152142129326129780047268409114441015993725554835256314039467401291,$
$x_G = 48439561293906451759052585252797914202762949526041747995844080717082404635286,$
$y_G = 36134250956749795798585127919587881956611106672985015071877198253568414405109,$
$x_H = 53987601597021778433910548064987973235945515666715026302948657055639179420355,$
$y_H = 53690949263410447908824456005055253553237881490194075871737490561466076234637.$

We give another point $H = (x_H, y_H)$ to pose $H = [k]G$ as an intractable instance of the ECDLP; this 256-bit prime field (and group order) is far beyond the reach of current attacks. For example, there is currently a campaign underway to solve a discrete logarithm problem over a 130-bit field using a cluster of servers that have already been running for two years (see `http://ecc-challenge.info/`), so (assuming the best known attacks stay exponential) it seems the above ECDLP should be safe for a while yet. We remark that the prime characteristic q is given by $q = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$; such primes are preferred in ECC as they allow for faster finite field multiplication and reduction routines, greatly enhancing the speed of $\mathbb{F}_q$ arithmetic. We now give a curve over the same field $\mathbb{F}_q$, for which the ECDLP is well within reach of the best known attacks. Namely, consider the alternative curve with $b = 0$, namely $\tilde{E}/\mathbb{F}_q : y^2 = x^3 - 3x$, whose group order $n = \#\tilde{E}$ is given as

$$\begin{aligned} n &= 115792089210356248762697446949407573530086143415290314195533631308867097853952, \\ &= 2^{96} \cdot 7 \cdot 274177 \cdot 67280421310721 \cdot 1131830892797394193140491 4103. \end{aligned}$$

This time, the largest prime divisor of the group order is only 94 bits long, and the complexity of solving the ECDLP in $\tilde{E}(\mathbb{F}_q)$ is governed by the difficulty of solving the ECDLP instance in this largest prime subgroup, which could be done in a small amount of time on a desktop computer.

The above example provides clear motivation as to the importance of counting points on elliptic curves. The largest prime factor of the group order determines the difficulty that attackers face when trying to solve the ECDLP, so we would like to be able to count points on curves quickly enough to find those whose order is prime or almost prime (i.e. has a small cofactor), or have methods of prescribing such a group order before searching for the curve. Fortunately, on elliptic curves we have efficient algorithms to do both.

We start our brief discussion on elliptic curve point counting by referring back to the two group orders in Example 2.1.15, and observing that both group orders share the first half of their digits with those of the field characteristic q. This suggests that the number of points on an elliptic curve is close to q, which is indeed the case in general; the *Hasse bound* [Sil09, Ch. 5, Th. 1.1] says the

most that $\#E(\mathbb{F}_q)$ can differ from $q+1$ is $2\sqrt{q}$, i.e. $|\#E(\mathbb{F}_q) - (q+1)| \leq 2\sqrt{q}$. This offset between $\#E(\mathbb{F}_q)$ and $(q+1)$ is called the *trace of Frobenius*, and is denoted by t, so

$$\#E(\mathbb{F}_q) = q + 1 - t, \qquad |t| \leq 2\sqrt{q} \tag{2.6}$$

We will discuss where t comes from and provide some more intuition behind the above formula in a moment, but what the Hasse bound tells us is that the group order lies somewhere in the interval $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$. In fact, Deuring [Deu41] showed that when q is prime[3], then every value $N \in [q+1-2\sqrt{q}, q+1+2\sqrt{q}]$ can be found as a group order $\#E(\mathbb{F}_q)$ for some E.

Example 2.1.16 (Magma script). Let $q = 23$, so that the Hasse interval becomes $[q+1-2\sqrt{q}, q+1+2\sqrt{q}] = [15, 33]$, meaning that there are exactly 19 different group orders taken by elliptic curves over $\mathbb{F}_{23}$. For example, $E/\mathbb{F}_{23} : y^2 = x^3 + 18x + 3$ has $\#E = 15$, whilst $\tilde{E}/\mathbb{F}_{23} : y^2 = x^3 + 13x + 7$ has $\#\tilde{E} = 33$. We give 19 (a, b) pairs such that the corresponding curves $E : y^2 = x^3 + ax + b$ have group orders in ascending order spanning the whole interval, as follows: $(18, 3)$, $(7, 22)$, $(19, 14)$, $(17, 17)$, $(12, 5)$, $(7, 12)$, $(8, 10)$, $(17, 18)$, $(20, 20)$, $(2, 3)$, $(20, 3)$, $(6, 8)$, $(16, 8)$, $(16, 22)$, $(9, 16)$, $(19, 6)$, $(20, 8)$, $(22, 9)$, $(13, 7)$.

A rough (but elementary and instinctive) argument as to why $\#E \approx q$ is that approximately half of the values $x \in [0, .., q-1]$ will give a quadratic residue $x^3 + ax + b \in \mathrm{QR}(q)$, which gives rise to two points $(x, \pm\sqrt{x^3 + ax + b}) \in E(\mathbb{F}_q)$, the only exception(s) being when $x^3 + ax + b = 0$ which obtains one point. The sophisticated explanation requires a deeper knowledge than our introduction offers, but for the purposes of this thesis we get almost all that we need from Equation (2.6); the derivation of which makes use of the following definition. If E is defined over $\mathbb{F}_q$, then the *Frobenius endomorphism* π is defined as

$$\pi : E \to E, \qquad (x, y) \mapsto (x^q, y^q). \tag{2.7}$$

We note that the Frobenius endomorphism maps any point in $E(\overline{\mathbb{F}}_q)$ to a point in $E(\overline{\mathbb{F}}_q)$, but the set of points fixed by π is exactly the group $E(\mathbb{F}_q)$. Thus, π only acts non-trivially on points in $E(\overline{\mathbb{F}}_q) \setminus E(\mathbb{F}_q)$, and more generally, $\pi^i : (x, y) \mapsto (x^{q^i}, y^{q^i})$ only acts non-trivially on points in $E(\overline{\mathbb{F}}_q) \setminus E(\mathbb{F}_{q^i})$.

[3]When q is a prime power, there are a very small number of explicitly described exceptions.

Example 2.1.17 (Magma script). Let $q = 67$, and consider $E/\mathbb{F}_q : y^2 = x^3+4x+3$, and let $\mathbb{F}_{q^2} = \mathbb{F}_q(u)$ where $u^2 + 1 = 0$, and further let $\mathbb{F}_{q^3} = \mathbb{F}_q(v)$ where $v^3 + 2 = 0$. For $P_1 = (15, 50) \in E(\mathbb{F}_q)$, we have $\pi_q(P_1) = (15^q, 50^q) = (15, 50)$. For $P_2 = (2u + 16, 30u + 39)$, we have $\pi_q(P_2) = ((2u + 16)^q, (30u + 39)^q) = (65u + 16, 39 + 37u)$; it is easy to see in this example that computing $\pi_q(Q)$ for any $Q \in E(\mathbb{F}_{q^2})$ involves a simple "complex conjugation" on each coordinate, which also agrees with $\pi_q^2(Q) = Q$. Let $P_3 = (15v^2 + 4v + 8, 44v^2 + 30v + 21)$, $\pi_q(P_3) = (33v^2+14v+8, 3v^2+38v+21)$, $\pi_q^2(P_3) = (19v^2+49v+8, 20v^2+66v+21)$, and $\pi_q^3(P_3) = P_3$.

We can now return to sketch the derivation of Equation (2.6) by skimming over results that are presented in full in Silverman's book [Sil09, Ch. V, Th. 1.1]. We now know that $P \in E(\mathbb{F}_q)$ if and only if $\pi(P) = P$ (i.e. $([1] - \pi)P = \mathcal{O}$), and thus $\#E(\mathbb{F}_q) = \#\ker([1] - \pi)$. It is not too hard to show that the map $[1] - \pi$ is separable, which means that $\#E(\mathbb{F}_q) = \#\ker([1] - \pi) = \deg([1] - \pi)$. We can then make use of (a special case of) a version of the Cauchy-Schwarz inequality [Sil09][Ch. V, Lemma 1.2], to give $|\deg([1] - \pi) - \deg([1]) - \deg(\pi)| \leq 2\sqrt{\deg([1])\deg(\pi)}$, from which Equation (2.6) follows from $\deg(\pi) = q$.

The theory of elliptic curves makes constant use of the *endomorphism ring* of E, denoted $\mathrm{End}(E)$, which (as the name suggests) is the ring of all maps from E to itself; addition in the ring is natural, i.e. $(\psi_1 + \psi_2)(P) = \psi_1(P) + \psi_2(P)$, and multiplication in $\mathrm{End}(E)$ is composition $(\psi_1\psi_2)(P) = \psi_1(\psi_2(P))$. The *multiplication-by-m* map $[m]$ is trivially in $\mathrm{End}(E)$ for all $m \in \mathbb{Z}$, and when E is defined over a finite field, then clearly π is too, so we are usually interested in any extra endomorphisms that shed more light on the behaviour of E.

Example 2.1.18 (Magma script). Consider $E/\mathbb{F}_q : y^2 = x^3+b$. The map ξ, defined by $\xi : (x, y) \mapsto (\xi_3 x, y)$ with $\xi_3^3 = 1$ and $\xi_3 \neq 1$, is a non-trivial endomorphism on E, so $\xi \in \mathrm{End}(E)$. If $\xi_3 \in \mathbb{F}_q$, then ξ will be defined over $\mathbb{F}_q$, otherwise $\xi_3 \in \mathbb{F}_{q^2}$ in which case ξ is not *defined over* $\mathbb{F}_q$, but over $\mathbb{F}_{q^2}$. We will observe both cases. Firstly, cubic roots of unity will be defined in $\mathbb{F}_q$ if and only if $q \equiv 1 \bmod 3$, so let us take $q \equiv 19$, $b = 5$, which gives $E/\mathbb{F}_{19} : y^2 = x^3 + 5$. Let $\xi_3 = 7$ so that $\xi_3^3 = 1$ (we could have also taken $\xi_3^2 = 11$), so that $\xi : (x, y) \mapsto (7x, y)$ is an endomorphism on E. Applying this to, say $P = (-1, 2)$, gives $\xi(P) = (-7, 2) \in E$. Taking the same curve over $\mathbb{F}_{23}$, i.e. $E/\mathbb{F}_{23} : y^2 = x^3+5$, for which $P = (-1, 2)$ is a again a point, we no longer have a non-trivial $\xi_3 \in \mathbb{F}_{23}$, so we must form a quadratic extension $\mathbb{F}_{q^2}(u)$, $u^2+1 = 0$. Now, we can take $\xi_3 = 8u+11$ (the other

option is $\xi_3^2 = 15u+11$), so that $\xi(P) = (-(8u+11), 2) = (15u+12, 2) \in E(\mathbb{F}_{q^2})$. Notice that P started in $E(\mathbb{F}_q)$, but landed in $E(\mathbb{F}_{q^2})$ under ξ. The endomorphism ξ has an inverse ξ^{-1} (which is defined the same way but with ξ_3^2 instead), so ξ is actually an automorphism of E, written as $\xi \in \mathrm{Aut}(E)$.

The definition of $\xi : (x, y) \mapsto (\xi_3 x, y)$ in the above example gives an endomorphism on $E : y^2 = x^3 + b$ regardless of the field that E is defined over. If there exists a non-trivial map (like ξ) for an elliptic curve E, we say E has *complex multiplication*. To be more precise, all elliptic curve endomorphism rings trivially contain $\mathbb{Z}$, since every $m \in \mathbb{Z}$ corresponds to the multiplication-by-m map $[m] \in \mathrm{End}(E)$. However, if non-trivial endomorphisms exist that make $\mathrm{End}(E)$ strictly larger than $\mathbb{Z}$, then we say E has complex multiplication (CM). Thus, by this definition, every elliptic curve defined over $\mathbb{F}_q$ has CM, because the existence of the Frobenius endomorphism $\pi \in \mathrm{End}(E)$ makes $\mathrm{End}(E)$ larger than $\mathbb{Z}$. However, if we discuss whether E has CM without yet stipulating the underlying finite field, then the question becomes non-trivial in general, because the answer depends on the existence of non-trivial maps. We use Silverman's example to illustrate [Sil09, Ch. 3, Eg. 4.4].

Example 2.1.19 (Magma script). Consider $E/K : y^2 = x^3 + ax$. The map $\zeta : (x, y) \mapsto (-x, iy)$, where $i^2 = -1$ in K is an endomorphism, so E has CM. Clearly, ζ will be defined over K if and only if $i \in K$. Observe that $\zeta \circ \zeta(x, y) = \zeta(-x, iy) = (x, -y) = -(x, y)$, so $\zeta \circ \zeta = [-1]$ (i.e. ζ^2 is equivalent to negation). Thus, there is a ring homomorphism $\mathbb{Z}[i] \to \mathrm{End}(E)$ defined by $m + ni \mapsto [m] + [n] \circ \zeta$. If $\mathrm{Char}(K) \neq 0$, then this map is an isomorphism, thus $\mathrm{End}(E) \cong \mathbb{Z}[i]$, and $\mathrm{Aut}(E) \cong \mathbb{Z}[i]^*$.

The trace of Frobenius t in Equation (2.6) is named so because of the role it plays in the characteristic polynomial satisfied by π, which is given as

$$\pi^2 - [t] \circ \pi + [q] = 0 \qquad \text{in } \mathrm{End}(E), \tag{2.8}$$

meaning that for all $(x, y) \in E(\overline{\mathbb{F}}_q)$, we have

$$(x^{q^2}, y^{q^2}) - [t](x^q, y^q) + [q](x, y) = \mathcal{O}. \tag{2.9}$$

Example 2.1.20 (Magma script). We use our results from Example 2.1.17 to illustrate, so as before $E/\mathbb{F}_{67} : y^2 = x^3 + 4x + 3$, $\mathbb{F}_{q^2} = \mathbb{F}_q(u)$ where $u^2 + 1 = 0$, and $\mathbb{F}_{q^3} = \mathbb{F}_q(v)$ where $v^3 + 2 = 0$. The trace of Frobenius is $t = -11$, so

$\#E(\mathbb{F}_q) = q + 1 - t = 79$. For $P_1 = (15, 50) \in E(\mathbb{F}_q)$, we trivially had $\pi^2(P_1) = \pi(P_1) = P_1$, so $P_1 - [t]P_1 + [q]P_1 = ([1] - [t] + [q])P_1 = [\#E(\mathbb{F}_q)]P_1 = \mathcal{O}$. For $P_2 = (2u+16, 30u+39)$, we had $\pi^2(P_2) = P_2$ and $\pi(P_2) = (65u+16, 37u+39)$, so we are computing $P_2 - [-11]\pi(P_2) + [67]P_2 = [68](2u+16, 30u+39) + [11](65u+16, 37u+39)$, which is indeed $\mathcal{O}$. $P_3 \in E(\mathbb{F}_{q^3})$ is the only case where both π and π^2 act non-trivially, so we compute $(19v^2+49v+8, 20v^2+66v+21) - [-11](33v^2+14v+8, 3v^2+38v+21) + [67](15v^2+4v+8, 44v^2+30v+21)$, which is $\mathcal{O}$.

We now give a brief sketch of Schoof's algorithm for counting points on elliptic curves [Sch85]. Understanding the algorithm is not a prerequisite for following this thesis, but it certainly warrants mention in any overview chapter on elliptic curves in cryptography, since it is essentially the algorithm that made ECC practical. Before Schoof's polynomial-time algorithm, all algorithms for point counting on elliptic curves were exponential and therefore cryptographically impractical. Besides, to sketch his idea, we need to introduce the notion of *division polynomials*, which are a useful tool in general. Put simply, division polynomials are polynomials whose roots reveal torsion points: namely, for odd[4] ℓ, the ℓ-th division polynomial $\psi_\ell(x)$ on E solves to give the x-coordinates of the points of order ℓ. They are defined recursively and depend on the curve constants a and b, but rather than giving the recursions here, we point the reader to [Sil09, Ch. III, Exer. 3.7], and opt instead for an example that illustrates their usefulness.

Example 2.1.21 (Magma script). Recall the curve $E/\mathbb{F}_{101} : y^2 = x^3 + x + 1$ from Example 2.1.13 with group order $\#E(\mathbb{F}_q) = 105 = 3 \cdot 5 \cdot 7$. The x-coordinates of the points of order 2 are found as the roots of $\psi_2(x) = 4x^3 + 4x + 4$, which is irreducible in $\mathbb{F}_q[x]$, so there are no 2-torsion points in $E(\mathbb{F}_q)$. For $r = 3$, $\psi_3(x) = 3x^4+6x^2+12x+100 \in \mathbb{F}_q[x]$ factors into $\psi_3(x) = (x+73)(x+84)(x^2+45x+36)$, so we get two solutions over $\mathbb{F}_q$, namely $x = 17$ and $x = 28$. This does not mean that the points implied by both solutions are in $\mathbb{F}_q$: namely, $x = 28$ gives $x^3 + x + 1 \in \mathrm{QR}(q)$, so two points in the 3-torsion follow as $(28, 8)$ and $(28, 93)$. Conversely, $x = 17$ gives $x^3 + x + 1 \notin \mathrm{QR}(q)$, so the two points implied by $x = 17$ will be defined over $\mathbb{F}_{q^2}$. For $\psi_5(x) = 5x^{12} + ... + 16$, the factorisation in $\mathbb{F}_q[x]$ is $\psi_5(x) = (x + 15)(x + 55)(x^5 + ... + 1)(x^5 + ... + 100)$, which gives $x = 46$ and $x = 86$ as solutions. This time, both x values give rise to two points, giving four non-trivial 5-torsion points in total: $(46, 25)$, $(46, 76)$, $(86, 34)$, $(86, 67)$. $\psi_7(x)$

[4]When ℓ is even, the division polynomial is of the form $\psi_\ell(x, y) = y \cdot \tilde{\psi}_\ell(x)$ since $y = 0$ gives points of order two, which are in the ℓ-torsion.

is degree 24, and gives three linear factors in $\mathbb{F}_q[x]$, all of which result in two 7-torsion points, giving 6 non-trivial torsion points in total: $(72, 5)$, $(72, 96)$, $(57, 57)$, $(57, 44)$, $(3, 43)$, $(3, 58)$. Other division polynomials have roots in $\mathbb{F}_q$, but these roots will not give rise to points defined over $\mathbb{F}_q$. For example, $\psi_{11}(x)$ has 5 roots over $\mathbb{F}_q$ (13, 18, 19, 22, 63), but none of them give points in $E(\mathbb{F}_q)$, meaning we will have to extend to $E(\mathbb{F}_{q^2})$ to collect any 11-torsion points. The only division polynomials whose roots produce points defined over $\mathbb{F}_q$ are the $\psi_d(x)$ with $d \mid 105$. This generalises to imply that the only division polynomials whose roots produce points defined over $\mathbb{F}_{q^n}$ are $\psi_d(x)$, where $d \mid \#E(\mathbb{F}_{q^n})$.

We are now in a position to shed light on Schoof's algorithm. Equation (2.6) means that computing $E(\mathbb{F}_q)$ immediately reduces to computing the (much smaller) trace of Frobenius, t. At the highest level, Schoof's idea is to compute $t_\ell \equiv t \bmod \ell$ for enough co-prime ℓ's to be able to uniquely determine t within the interval $-2\sqrt{q} \leq t \leq 2\sqrt{q}$ via the Chinese Remainder Theorem. Namely, when $\prod_\ell t_\ell \geq 4\sqrt{q}$, then we have enough relations to determine the correct t. To compute t_ℓ for various primes ℓ, Schoof looked to consider Equation (2.9) "modulo ℓ", restricting the points (x, y) to come from the ℓ-torsion, and trying to solve

$$(x^{q^2}, y^{q^2}) - [t_\ell](x^q, y^q) + [q_\ell](x, y) = \mathcal{O}, \tag{2.10}$$

for t_ℓ, where $q_\ell \equiv q \bmod \ell$. The problem for general ℓ is, that since we do not know the group order, we cannot explicitly use ℓ-torsion points in (2.10), nor do we know if they are even defined over $\mathbb{F}_q$, or where they *are* defined, so we have to work with (2.10) implicitly. Namely, we restrict (2.10) to the ℓ-torsion by working modulo $\psi_\ell(x)$: we do not work with Equation (2.10) on $E(\mathbb{F}_q)$, but rather in the polynomial ring $R_\ell = \mathbb{F}_q[x, y]/\langle \psi_\ell(x), y^2 - (x^3 + ax + b)\rangle$, where the size of the polynomials $f(x, y)$ we deal with in R_ℓ are bounded by the degrees of the division polynomials $\psi_\ell(x)$. Even for very large prime fields $\mathbb{F}_q$ of cryptographic size, the number of different primes used is small enough to keep this algorithm very practical. For example, finding the group order of the curve defined over a 256-bit prime q in Example 2.1.15 would require solving (2.10) for the 27 primes up to $\ell = 107$, at which point the product of all the primes used exceeds $4\sqrt{q}$. It is not too difficult to deduce that the asymptotic complexity of Schoof's algorithm is $O\left((\log q)^8\right)$ (see [Sil09, Ch. XI.3] for details, and further improvements).

Example 2.1.22 (Magma script). Consider $E/\mathbb{F}_{13} : y^2 = x^3 + 2x + 1$; we seek

$\#E(\mathbb{F}_{13})$. Schoof's algorithm actually begins with $\ell = 3$ [Sil09, Ch. XI.3]; so since $14 < 4\sqrt{13} < 15$, we only need to solve (2.10) with $\ell = 3$ and $\ell = 5$. For $\ell = 3$, $\psi_3(x) = 3x^4 + 12x^2 + 12x + 9$, so we work in the ring $R_3 = \mathbb{F}_q[x,y]/\langle 3x^4 + 12x^2 + 12x + 9, y^2 - (x^3 + 2x + 1)\rangle$ with $q_\ell = 1$, to find that $t_3 = 0$. For $\ell = 5$, $\psi_5(x) = 5x^{12} + ... + 6x + 7$, so we work in the ring $R_5 = \mathbb{F}_q[x,y]/\langle 5x^{12} + ... + 6x + 7, y^2 - (x^3 + 2x + 1)\rangle$ with $q_\ell = 3$ to find that $t_5 = 1$. For both cases we had to compute $[q_\ell](x,y)$ in R_ℓ using the affine formulas (2.4) and (2.5), compute (x^q, y^q) and (x^{q^2}, y^{q^2}) in R_ℓ, and then test incremental values of t_ℓ until $[t_\ell](x^q, y^q)$ (also computed with the affine formulas) satisfies (2.10). The CRT with $t \equiv 0 \bmod 3$ and $t \equiv 1 \bmod 5$ gives $t \equiv 6 \bmod 15$, which combined with $-7 \leq t \leq 7$ means $t = 6$, giving $\#E = q + 1 - t = 8$.

We finish this section by briefly discussing one more improvement to ECC that will essentially bring the reader up to speed with major milestones that contribute to the current state-of-the-art implementations. The technique was introduced by Gallant, Lambert and Vanstone (GLV) [GLV01], and recently generalised by Galbraith, Lin and Scott (GLS) [GLS11]. It exploits the existence of an efficiently computable endomorphism ψ that allows us to instantly move P to a large multiple $\psi(P) = [\lambda]P$ of itself, so that (in the simplest case) the scalar multiplication $[m]P$ can be split into $[m]P = [m_0]P + [m_1]\psi(P)$, where if $|m| \approx r$ (the large subgroup order), then $|m_0|$, $|m_1| \approx \sqrt{r}$. The values m_0 and m_1 are found by solving a closest vector problem in a lattice [GLV01, §4]. We apply an example from the GLV paper (which was itself taken from Cohen's book [Coh96, §7.2.3]) that is actually exploiting a special case of the endomorphism we described in Example 2.1.19.

Example 2.1.23 (Magma script). Let $q \equiv 1 \bmod 4$ be prime, $E/\mathbb{F}_q : y^2 = x^3 + ax$, and let $i^2 = -1$. The map defined by $\psi : (x,y) \mapsto (-x, iy)$ and $\psi : \mathcal{O} \mapsto \mathcal{O}$ is an endomorphism defined over $\mathbb{F}_q$ ($\psi = \zeta$ from 2.1.19). Let $P \in E(\mathbb{F}_q)$ have prime order r, then $\psi(Q) = [\lambda]Q$ for all $Q \in \langle P\rangle$, and λ is the integer satisfying $\lambda^2 = -1 \bmod r$. We give a specific example: $q = 1048589$, $E/\mathbb{F}_q : y^2 = x^3 + 2x$ with $\#E = 2r$, where $r = 524053$; we further have $i = 38993$, and $\lambda = 304425$. $P = (609782, 274272) \in E$ has $|\langle P\rangle| = r$, so we can take any element in $\langle P\rangle$, say $Q = (447259, 319154)$, and compute $\psi(Q) = (-447259, i \cdot 319154) = (601330, 117670) = [304425](447259, 319154) = [\lambda]Q$. Computing a random multiple of Q, say $[m]Q$ with $m = 103803$, can be done by decomposing m into (in this case) $(m_0, m_1) = (509, 262)$, and instead computing $[m]Q =$

$[m_0]Q+[m_1]\psi(Q)$. Here m is 17 bits, whilst m_0 and m_1 are both 9 bits. Doing the scalar multiples $[m_0]Q$ and $[m_1]\psi(Q)$ separately would therefore give no savings, but where the GLV/GLS methods gain a substantial speed-up is in merging the doublings required in both of the multiplications by the "mini-scalars", which halves the number of doublings required overall; again, see [GLV01, GLS11] for futher details.

2.1.3 Section summary

We defined the elliptic curve group law $\oplus$ via the chord-and-tangent method, and discussed that elliptic curve groups are an attractive setting for discrete-log based cryptosystems because of the relative security obtained for the sizes of the fields they are defined over. We also exemplified many improvements in the context of cryptographic implementations, where the fundamental operation (that creates ECDLP instances) is computing large scalar multiples $[m]P$ of $P \in E$. Namely, we showed that group law computations in finite fields can be much faster in projective coordinates, i.e. computing $(X_1 : Y_1 : Z_1) \oplus (X_2 : Y_2 : Z_2)$ rather than $(x_1, y_1) \oplus (x_2, y_2)$, and that other (non-Weierstrass) curve models also offer advantages. We gave an explicit equation for the number of points in $E(\mathbb{F}_q)$, and briefly discussed Schoof's polynomial-time algorithm that facilitates point counting on curves of cryptographic size. We also introduced the notion of the endomorphism ring $\mathrm{End}(E)$ of E, and finished by showing that non-trivial elements of $\mathrm{End}(E)$ can be used to further accelerate ECC. A reader that is comfortable with the exposition in this section is equipped with many of the tools required to tackle the vast literature in this field, and is somewhat up-to-date with the state-of-the-art ECC implementations. For example, in the context of chasing ECC speed records, some authors have applied alternative projective coordinate systems to the Edwards model to give very fast scalar multiplications [HWCD08], whilst others have investigated higher dimension GLV/GLS techniques (Example 2.1.23 above was 2-dimensional) to gain big speed-ups [HLX12]; visit `http://bench.cr.yp.to/supercop.html` for comprehensive and up-to-date benchmarkings of a wide number of implementations that are pushing ECC primitives to the limit.

Relaxed notation. Our last order of business before proceeding into the next

section is to relax some notation in order to agree with the rest of the literature. Rather than writing "$\oplus$" for the elliptic curve group law, from hereon we simply use "+". Similarly, for the inverse of the point P, we use $-P$ instead of $\ominus P$.

2.2 Divisors

In this section we introduce some basic language and definitions from algebraic geometry that are fundamental to the understanding of cryptographic pairing computations. We continue with our example-driven approach and illustrate each concept and definition as it arises. We will essentially just be expanding on the more concise section found in Galbraith's chapter [Gal05, §IX.2]. However, we only focus on what we need in this thesis (to describe elliptic curve pairings), so we refer any reader seeking a more general and thorough treatment to Galbraith's new book [Gal12, Ch.7-9]. Since our exposition targets the newcomer, we begin by assuring such a reader that their persistence through the definitions and examples will be amply rewarded. On becoming comfortable with the language of divisors, one can immediately start to appreciate how pieces of the "pairings puzzle" fit together very naturally, and might even enjoy feeling intuition behind important theorems that would otherwise appear foreign.

The following statements apply to all curves C over any perfect field K and its closure $\overline{K}$ (see [Sil09, p. 17, p. 1] for the respective definitions). However, for now we place the discussion in our context and specialise to the case where C is an elliptic curve E over a finite field $K = \mathbb{F}_q$. Later in this section we will expand to more general examples and statements in time to present the important theorems in their full generality. A *divisor* D on E is a convenient way to denote a multi-set of points on E, written as the formal sum

$$D = \sum_{P \in E(\overline{\mathbb{F}}_q)} n_P(P),$$

where all but finitely many $n_P \in \mathbb{Z}$ are zero. The standard parentheses $(\cdot)$ around the P's and the absence of square parentheses $[\cdot]$ around the n_P's is what differentiates the formal sum in a divisor from an actual sum of points (i.e. using the group law) on E. The set of all divisors on E is denoted by $\mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$ and forms a group, where addition of divisors is natural, and the identity is the divisor with all $n_P = 0$, the zero divisor $0 \in \mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$. The *degree* of a divisor

D is $\mathrm{Deg}(D) = \sum_{P \in E(\overline{\mathbb{F}}_q)} n_P$, and the *support* of D, denoted $\mathrm{supp}(D)$, is the set $\mathrm{supp}(D) = \{P \in E(\overline{\mathbb{F}}_q) : n_P \neq 0\}$.

Example 2.2.1 (Magma script). Let $P, Q, R, S \in E(\overline{\mathbb{F}}_q)$. Let $D_1 = 2(P) - 3(Q)$, and $D_2 = 3(Q) + (R) - (S)$, so that $\mathrm{Deg}(D_1) = 2 - 3 = -1$, and $\mathrm{Deg}(D_2) = 3+1-1 = 3$. The sum $D_1+D_2 = 2(P)+(R)-(S)$, and naturally $\mathrm{Deg}(D_1+D_2) = \mathrm{Deg}(D_1) + \mathrm{Deg}(D_2) = 2$. The supports are $\mathrm{supp}(D_1) = \{P, Q\}$, $\mathrm{supp}(D_2) = \{Q, R, S\}$, and $\mathrm{supp}(D_1 + D_2) = \{P, R, S\}$.

Associating divisors with a function f on E is a convenient way to write down the intersection points (and their multiplicities) of f and E. Let $\mathrm{ord}_P(f)$ count the multiplicity of f at P, which is positive if f has a zero at P, and negative if f has a pole at P. We write the *divisor of a function* f as (f), and it is defined as the divisor

$$(f) = \sum_{P \in E(\overline{\mathbb{F}}_q)} \mathrm{ord}_P(f)(P).$$

Example 2.2.2 (Magma script). We have already seen examples of functions on E in the previous section, namely the lines $\ell : y = \lambda x + \nu$ used in the chord-and-tangent rule, and it is natural that we are really only interested in the points of intersection of ℓ and E, which is exactly what the divisor (ℓ) tells us. The chord ℓ in Figure 2.14 intersects E in P, Q and $-(P + Q)$, all with

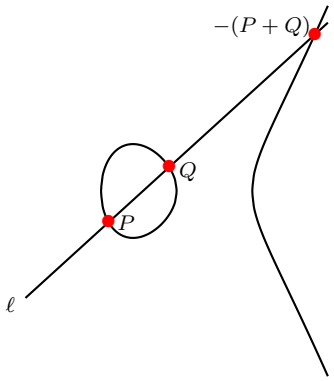


Figure 2.14: $(\ell) = (P) + (Q) + (-(P + Q)) - 3(\mathcal{O})$.

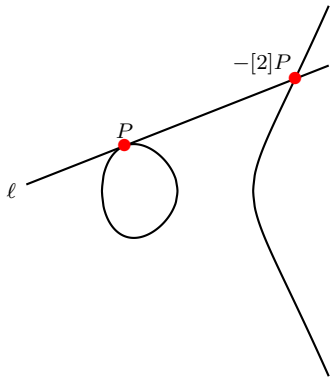


Figure 2.15: $(\ell) = 2(P)+(-[2]P)-3(\mathcal{O})$.

multiplicity 1, and (as we will discuss further in a moment) ℓ also intersects E with multiplicity -3 at $\mathcal{O}$, i.e. ℓ has a pole of order 3 at $\mathcal{O}$. Thus, ℓ has divisor $(\ell) = (P) + (Q) + (-(P + Q)) - 3(\mathcal{O})$. The tangent ℓ in Figure 2.15 intersects E

with multiplicity 2 at P, with multiplicity 1 at $-[2]P$, and again with multiplicity -3 at $\mathcal{O}$, so in this case $(\ell) = 2(P) + (-[2]P) - 3(\mathcal{O})$. Notice that in both cases we have $\mathrm{Deg}\,((\ell)) = 0$.

The balance that occurred between the zeros and poles in Example 2.2.2 that led to $\mathrm{Deg}((\ell)) = 0$ is not a coincidence. In fact, a fundamental result that lies at the heart of the discussion is that this always happens: namely, for any function f on E, we always have $\mathrm{Deg}((f)) = 0$. An instructive proof of this result is in Galbraith's book [Gal12, Th. 7.7.1], but roughly speaking this property follows from observing that the degree of the affine equation that solves for the zeros of f on E matches the degree of the projective equation that determines the multiplicity of the pole of f at $\mathcal{O}$, i.e. the projective version of f is g/h where g and h both have the same degree as f. We revisit Example 2.2.2 and illustrate in this special case.

Example 2.2.3 (Magma script). We already know that three zeros (counting multiplicities) will always arise from substituting $\ell : y = \lambda x + \nu$ into $E/\mathbb{F}_q : y^2 = x^3 + ax + b$, but we have only considered ℓ on the affine curve $E \cap \mathbb{A}^2$, where ℓ has no poles. To consider ℓ on E at $\mathcal{O} = (0 : 1 : 0)$ (in $\mathbb{P}^2(\mathbb{F}_q)$), we need to take $x = X/Z$ and $y = Y/Z$ which gives $(\frac{\lambda X+\nu Z}{Z})^2 = (\frac{X}{Z})^3 + a(\frac{X}{Z}) + b$, for which we clearly have a pole of order 3 when $Z = 0$.

The algebra between functions naturally translates across to the algebra between their divisors, so $(fg) = (f) + (g)$ and $(f/g) = (f) - (g)$, $(f) = 0$ if and only if f is constant, and thus if $(f) = (g)$, then $(f/g) = 0$ so f is a constant multiple of g, which means that the divisor (f) determines f up to non-zero scalar multiples.

Example 2.2.4 (Magma script). Let $\ell : y = \lambda_1 x + \nu_1$ be the chord (through P and Q) with divisor $(\ell) = (P) + (Q) + (-(P + Q)) - 3(\mathcal{O})$, and let $\ell' : y = \lambda_2 x + \nu_2$ be the tangent at R with divisor $(\ell') = 2(R) + (-[2]R) - 3(\mathcal{O})$. The divisor of the function $\ell_{\mathrm{prod}} = \ell\ell'$ is $(\ell_{\mathrm{prod}}) = (\ell) + (\ell') = (P) + (Q) + 2(R) + (-(P + Q)) + (-[2]R) - 6(\mathcal{O})$. The divisor of $\ell_{\mathrm{quot}} = \ell/\ell'$ is $(\ell_{\mathrm{quot}}) = (\ell) - (\ell') = (P) + (Q) + (-(P + Q)) - 2(R) - (-[2]R)$. Notice that ℓ_{quot} does not intersect E at $\mathcal{O}$; projectifying $\ell/\ell' = \frac{y-\lambda_1 x+\nu_1}{y-\lambda_2 x+\nu_2}$ gives $\frac{Y-\lambda_1 X+\nu_1 Z}{Y-\lambda_2 X+\nu_2 Z}$, which does not give rise to any zeros or poles at $Z = 0$. Suppose we wanted to depict the function $\ell\ell'$ on E, and we multiplied out $(y - \lambda_1 x - \nu_1)(y - \lambda_2 x - \nu_2)$, substituted the y^2 for $x^3 + ax + b$ and wrote $y = \frac{x^3+ax+b+(\lambda_1 x+\nu_1)(\lambda_2 x+\nu_2)}{(\lambda_1+\lambda_2)x+\nu_1+\nu_2}$. It does not make sense to try and depict this function since all the pictures we have used for illustrative

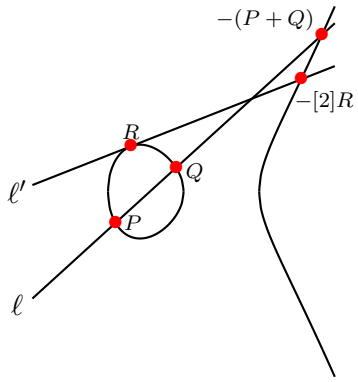


Figure 2.16: Two functions ℓ and ℓ' on E.

purposes also show how the functions (on E) behave at points that are not on E, where the substitution $y^2 = x^3 + ax + b$ is not permitted.

2.2.1 The divisor class group

We can now start introducing important subgroups of the group of divisors $\mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$ on E. We temporarily drop the subscript, and write $\mathrm{Div}(E)$ as the group of all divisors on E. The set of degree zero divisors $\{D \in \mathrm{Div}(E) : \mathrm{Deg}(D) = 0\}$ forms a proper subgroup, which we write as $\mathrm{Div}^0(E) \subset \mathrm{Div}(E)$. If a divisor D on E is equal to the divisor of a function, i.e. $D = (f)$, then D is called a *principal* divisor, and the set of principal divisors naturally form a group, written as $\mathrm{Prin}(E)$. We already know (from Example 2.2.3 and the preceding discussion) that principal divisors have degree zero, but there are also degree zero divisors that are not the divisors of a function, so the degree zero subgroup is strictly larger than the principal divisors, i.e. $\mathrm{Prin}(E) \subset \mathrm{Div}^0(E)$. There is, however, an extra condition on elements of $\mathrm{Div}^0(E)$ that *does* allow us to write an "if-and-only-if": $D = \sum_P n_P(P) \in \mathrm{Div}^0(E)$ is principal if and only if $\sum_P [n_P]P = \mathcal{O}$ on E [Gal05, Th. IX.2]. We illustrate this statement, and the relationship between the three groups

$$\mathrm{Prin}(E) \subset \mathrm{Div}^0(E) \subset \mathrm{Div}(E) \tag{2.11}$$

in Example 2.2.5.

Example 2.2.5 (Magma script). Consider $E/\mathbb{F}_{103} : y^2 = x^3 + 20x + 20$, with points $P = (26, 20)$, $Q = (63, 78)$, $R = (59, 95)$, $S = (24, 25)$, $T = (77, 84)$,

$U = (30, 99)$ all on E. The divisor $(S) + (T) - (P) \in \mathrm{Div}(E)$ is clearly not in the subgroup $\mathrm{Div}^0(E)$, since it has degree 1; there are also infinitely many other trivial examples. The divisor $(P) + (Q) - (R) - (S)$ is in $\mathrm{Div}^0(E)$, but is not principal since $P + Q - R - S = (18, 49) \neq \mathcal{O}$ on E. Thus, a function f with $(f) = (P) + (Q) - (R) - (S)$ does not exist. On the other hand, the divisor $(P) + (Q) - (R) - (T)$ is principal, since it is degree 0 and $P + Q - R - T = \mathcal{O}$ on E. Thus, there is some function f on E such that $(f) = (P) + (Q) - (R) - (T)$; it is $f = \frac{6y+71x^2+91x+91}{x^2+70x+11}$. The sum $R+T$ on E is actually U, thus $P+Q-U = \mathcal{O}$ on E, but this time there is no function with divisor $(P) + (Q) - (U)$ because the degree of this divisor is not zero; however, we can keep the sum on E as $\mathcal{O}$ but manipulate the degree by instead taking the divisor $(P) + (Q) - (U) - (\mathcal{O})$, which must be in $\mathrm{Prin}(C)$, guaranteeing the existence of a function g with $(g) = (P)+(Q)-(U)-(\mathcal{O})$, namely $g = \frac{y+4x+82}{x+73}$. Observe the difference between f and g in projective space, where $f = \frac{6YZ+71X^2+91XZ+91Z^2}{X^2+70XZ+11Z^2}$ and $g = \frac{Y+4X+82Z}{X+73Z}$. For f, the point at infinity $\mathcal{O} = (0 : 1 : 0)$ zeros both the numerator and denominator, giving a zero and a pole which cancels out its contribution to (f), whilst for g, the point at infinity only zeros the denominator, which is why $\mathcal{O} \in \mathrm{supp}((g))$, whereas $\mathcal{O} \notin \mathrm{supp}((f))$.

Returning to the subscript notation for a moment, the three subgroups (and other related groups) in Equation (2.11) are often accompanied by the field they apply to, e.g. for a general field K, they are written as $\mathrm{Prin}_K(E)$, $\mathrm{Div}^0_K(E)$, and $\mathrm{Div}_K(E)$. Here $\mathrm{Div}_K(E) \subset \mathrm{Div}(E)$ is formally defined as the set of divisors invariant under the action of $\mathrm{Gal}(\overline{K}/K)$, where $\sigma \in \mathrm{Gal}(\overline{K}/K)$ acts on $D = \sum_P n_P(P)$ to give $D^\sigma = \sum_P n_P(\sigma(P))$, so that $D \in \mathrm{Div}_K(E)$ if $D = D^\sigma$. This is very natural in the contexts we consider, so we will continue on without subscripts.

Before we define the *divisor class group* of E, we look at the important notion of divisor equivalence in $\mathrm{Div}(E)$. We call the divisors D_1 and D_2 *equivalent*, written as $D_1 \sim D_2$, if $D_1 = D_2 + (f)$ for some function f.

Example 2.2.6 (Magma script). Consider $P = (57, 24)$, $Q = (25, 37)$, $R = (17, 32)$ and $S = (42, 35)$ on $E/\mathbb{F}_{61} : y^2 = x^3 + 8x + 1$. The divisors $D_1 = (P) + (Q) + (R)$ and $D_2 = 4(\mathcal{O}) - (S)$ are equivalent as follows. The function $f : y = 33x^2 + 10x + 24$, which intersects E at P, Q, R and S with multiplicity 1, and therefore has a pole of order 4 at infinity, has divisor $(f) = (P) + (Q) + (R) + (S) - 4(\mathcal{O})$, meaning $D_1 = D_2 + (f)$, so $D_1 \sim D_2$. Alternatively, if we did

not want to find f, we could have used $D_1 - D_2 = (P) + (Q) + (R) + (S) - 4(\mathcal{O})$, which has degree zero, and computed that $P + Q + R + S - [4]\mathcal{O} = \mathcal{O}$ on E, which means $D_1 - D_2 \in \mathrm{Prin}(E)$, so that $D_1 - D_2 = (f)$ for some function f.

The *divisor class group*, or *Picard group*, of E is defined as the quotient group

$$\mathrm{Pic}^0(E) = \mathrm{Div}^0(E)/\mathrm{Prin}(E), \tag{2.12}$$

i.e. the divisor class group is the group of all degree zero divisors modulo the principal divisors on E. At first read, this notion of equivalence (modulo divisors of functions) may seem a little abstract, but once we see it in action (particularly in more general scenarios than elliptic curves), it becomes very natural. We will first use this notion to describe the elliptic curve group law in terms of divisors, following along the lines of Galbraith [Gal05, §IX.2].

Example 2.2.7 (Magma script). Referring back to Figure 2.5 (or Figure 2.6 in the case that $Q = P$), the line ℓ joining P and Q has divisor $(\ell) = (P)+(Q)+(-R)-3(\mathcal{O})$, whilst the vertical line $v = x - x_R$ has divisor $(v) = (-R) + (R) - 2(\mathcal{O})$. The quotient $\frac{\ell}{v}$ has divisor $(\frac{\ell}{v}) = (P) + (Q) - (R) - (\mathcal{O})$. Thus, the equation $R = P + Q$ on E is the same as the divisor equality $(R) - (\mathcal{O}) = (P) - (\mathcal{O}) + (Q) - (\mathcal{O}) - (\frac{\ell}{v})$, and the map of points to divisor classes $P \mapsto (P) - (\mathcal{O})$ is a group homomorphism. To concretely connect this back to Equation (2.12), both $(R) - (\mathcal{O})$ and $(P) + (Q) - 2(\mathcal{O})$ are clearly in $\mathrm{Div}^0(E)$, but they represent the same class in $\mathrm{Pic}^0(E)$, because the divisor $(\frac{l}{v}) = (P) + (Q) - (R) - (\mathcal{O})$ (which is their difference) is principal, and therefore zero in $\mathrm{Pic}^0(E)$.

2.2.2 A consequence of the Riemann-Roch Theorem

The notion of equivalence allows us to *reduce* divisors of any size $D \in \mathrm{Pic}^0(E)$ into much smaller divisors. We will make this statement precise after an example, but we must first define what we mean by "size". A divisor $D = \sum_P n_P(P)$ is called *effective* if $n_P \geq 0$ for all $P \in E$. The only divisor in $\mathrm{Div}^0(E)$ that is effective is the zero divisor. Thus, we define the *effective part* of a divisor D as $\epsilon(D) = \sum_P n_P(P)$, where $n_P \geq 0$. For example, the divisor $D = (P)+(Q)-2(\mathcal{O})$ is not effective, but the effective part is $\epsilon(D) = (P) + (Q)$. By the *size* of D, we mean the degree of the effective part, so in our example, although $\mathrm{Deg}(D) = 0$, it is size 2, since $\mathrm{Deg}(\epsilon(D)) = 2$.

Example 2.2.8 (Magma script). Consider the divisor $D = (P_1)+...+(P_{11})-11(\mathcal{O})$ (with $\mathrm{Deg}(\epsilon(D)) = 11$) as an element of $\mathrm{Pic}^0(E)$ on $E/\mathbb{F}_q : y^2 = x^3 + ax + b$, where the P_i are not necessarily distinct. To find a divisor that is equivalent to D, we can construct function $\ell_{10} : y = a_{10}x^{10} + ... + a_1x + a_0$ to interpolate the distinct points in $\mathrm{supp}(D)$ with appropriate multiplicities. Substituting ℓ_{10} into E gives a degree 20 polynomial in x, the roots of which reveals the 20 affine points of intersection (counting multiplicities) between ℓ_{10} and E. We already know 11 of these points (the P_i's), so let $P'_1, ...P'_9$ be the other 9. An important point to note is that these points are not necessarily defined over $\mathbb{F}_q$. Since $(\ell_{10}) = \sum_{i=1}^{11}(P_i) + \sum_{i=1}^{9}(P'_i) - 20(\mathcal{O}) \in \mathrm{Prin}(E)$, $D' = -(\sum_{i=1}^{9}(P'_i) - 9(\mathcal{O}))$ is a divisor equivalent to D in $\mathrm{Pic}^0(E)$, i.e. $D' \sim D$. We can repeat this process, interpolating the points in $\mathrm{supp}(D')$ with a degree 8 polynomial $\ell_8 : y = a'_8x^8 + ... + a'_1x + a'_0$, which will intersect E (in the affine sense) 16 times, giving 7 new intersection points, thereby finding a divisor $D'' = \sum_{i=1}^{7}(P''_i)-7(\mathcal{O})$ equivalent to D', meaning $D'' \sim D$. It is easy to infer that the number of new roots (maximum number of divisors in the consecutive supports) decreases each time by two, so that in two more steps we will arrive at $\tilde{D} = (\tilde{P}_1)+(\tilde{P}_2)+(\tilde{P}_3)-3(\mathcal{O})$. We can interpolate the three points in $\mathrm{supp}(\tilde{D})$ with a quadratic function $\tilde{\ell} : y = \tilde{a}_2x^2 + \tilde{a}_1x + \tilde{a}_0$ that clearly intersects E at one more affine point, say Q. That is, $(\tilde{\ell}) = (\tilde{P}_1) + (\tilde{P}_2) + (\tilde{P}_3) + (Q) - 4(\mathcal{O})$, and since $(\tilde{\ell}) \in \mathrm{Prin}(E)$, then

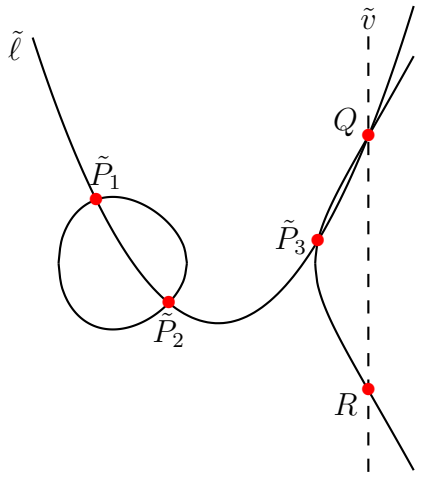


Figure 2.17: Reducing $\tilde{D}$ to $(R) - (\mathcal{O})$ in $\mathrm{Pic}^0(E)$.

$(\tilde{D}) \sim (\mathcal{O}) - (Q)$. Lastly, the vertical line $\tilde{v}$ has divisor $(\tilde{v}) = (Q) + (R) - 2(\mathcal{O})$, meaning $(\mathcal{O}) - (Q) \sim (R) - (\mathcal{O})$, which gives $(\tilde{D}) \sim (R) - (\mathcal{O})$. To summarise, we started with a divisor $D = (P_1) + ...(P_{11}) - 11(\mathcal{O})$ which had size 11, and

reduced to the equivalent divisor $(R) - (\mathcal{O}) \sim D$ in $\mathrm{Pic}^0(E)$ which has size 1.

The above example illustrates a key consequence of one of the most central theorems in our study: the *Riemann-Roch* theorem. To present the theorem in its generality requires a few more definitions than we need for our exposition, so for the full story we refer the reader to any of [Ful08, §8], [Sil09, §II.5], [Gal12, §8.7]. The important corollary we use is the following: for any curve C, there is a unique minimal integer g, called the *genus* of C, such that any divisor $D \in \mathrm{Pic}^0(C)$ is equivalent to a divisor D' with $\mathrm{Deg}(\epsilon(D')) \leq g$. Elliptic curves E are curves of genus $g = 1$, meaning that every $D \in \mathrm{Pic}^0(E)$ can be written as $(P_1) - (Q_1)$; this is why we were able to *reduce* the divisor in Example 2.2.8 to $(R) - (\mathcal{O})$.

We will only be dealing with hyperelliptic curves in this thesis, since they have proved most successful in cryptography. We will not be discussing them until Chapter 7, but for now it aids one's understanding to see where elliptic curves fit in a slightly broader context. Assuming an odd characteristic field, a general ("imaginary quadratic") hyperelliptic curve of genus g is a generalisation of an elliptic curve, which can be written as

$$C_g : y^2 = x^{2g+1} + f_{2g}x^{2g} + ... + f_1x + f_0. \tag{2.13}$$

Each divisor $D \in \mathrm{Pic}^0(C_g)$ has a unique *reduced* representative of the form

$$(P_1) + (P_2) + ... + (P_n) - n(\mathcal{O}),$$

where $n \leq g$, $P_i \neq -P_j$ for all $i \neq j$, and no P_i satisfying $P_i = -P_i$ appears more than once [BBC+09, §2.3]. The following examples illustrate this in the case of genus 2 and genus 3 respectively.

Example 2.2.9 (Magma script). A general (odd characteristic field) hyperelliptic curve of genus $g = 2$ is given (via Equation (2.13)) as $C_2 : y^2 = x^5 + f_4x^4 + ... + f_0$; we give a typical depiction in Figure 2.18. Suppose we have a divisor $D = (P_1) + (P_2) + (P_3) + (P_4) - 4(\mathcal{O}) \in \mathrm{Pic}^0(C_2)$, the affine support of which is depicted in red.

The Riemann-Roch theorem guarantees a (unique) equivalent divisor of the form $(P_1') + (P_2') - 2(\mathcal{O})$. We find it by constructing the cubic function $\ell : y = a_3x^3 + ... + a_0$ that has 4 zeros corresponding to the effective part of D, and therefore 4 poles at $\mathcal{O}$. Substitution of ℓ into E reveals two more points of intersection, $\bar{P}_1$ and $\bar{P}_2$, meaning $(\ell) = (P_1)+(P_2)+(P_3)+(P_4)+(\bar{P}_1)+(\bar{P}_2)-6(\mathcal{O})$.

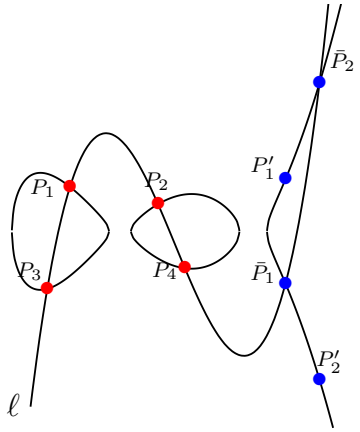


Figure 2.18: Reducing $D = \sum_{i=1}^{4}((P_i) - (\mathcal{O}))$ to $D' = \sum_{i=1}^{2}((P_i') - (\mathcal{O})) \sim D$.

Since $(\ell) \in \text{Prin}(C_2)$, then $D = D - (\ell)$ in $\text{Pic}^0(C_2)$ meaning $D \sim 2(\mathcal{O}) - (\bar{P}_1) - (\bar{P}_2)$. As usual, we reverse the ordering (so the effective part is affine) by making use of the vertical lines v_1 and v_2 with divisors $(v_1) = (\bar{P}_1) + (P_1') - 2(\mathcal{O})$ and $(v_2) = (\bar{P}_2) + (P_2') - 2(\mathcal{O})$, to write $2(\mathcal{O}) - (\bar{P}_1) - (\bar{P}_2) = 2(\mathcal{O}) - (\bar{P}_1) - (\bar{P}_2) + (v_1) + (v_2) = (P_1') + (P_2') - 2(\mathcal{O}) = D'$, meaning $D \sim D'$. We have reduced a divisor D with $\text{Deg}(\epsilon(D)) = 4$ to a divisor D' with $\text{Deg}(\epsilon(D')) = 2 \leq g$. Note that the points in the support of D' are not necessarily defined over $\mathbb{F}_q$. Also note that trying to reduce D' any further, say by running a line $\ell' : y = \lambda x + \nu$ through P_1' and P_2', will not work in general, since this line will intersect E in 3 more places, creating an unreduced divisor D'' with $\text{Deg}(\epsilon(D'')) = 3 > g$.

Example 2.2.10 (Magma script). Consider a general genus 3 hyperelliptic curve $C_3 : y^2 = x^7 + f_6x^6 + ... + f_0$; a typical depiction is given in Figure 2.19, with a vertically magnified Figure version in 2.20. Consider the divisor $D = \sum_{i=1}^{6}((P_i) - (\mathcal{O})) \in \text{Pic}^0(C_3)$, the affine support of which is the red points in Figure 2.19.

We reduce D by determining the other points of intersection between the quintic interpolator $\ell : y = a_5x^5 + \ + a_0$ and C_3, of which there are 4: $\bar{P}_1, ..., \bar{P}_4$ depicted in green on C_3. $(\ell) = 0$ in the divisor class group so $\sum_{i=1}^{6}((P_i) - (\mathcal{O})) + \sum_{i=1}^{4}((\bar{P}_i) - (\mathcal{O})) = 0$, but the degree of the effective part of $\sum_{i=1}^{4}((\bar{P}_i) - (\mathcal{O}))$ is still larger than g, so obtaining the unique reduced divisor requires further reduction. Namely, the cubic function $\bar{\ell} : y = \bar{a}_3x^3 + ... + \bar{a}_0$ (depicted in green) interpolates the four green points and (when substituted into C_3) clearly intersects C_3 in another 3 affine points, depicted in blue. Thus, $\sum_{i=1}^{4}((\bar{P}_i) - (\mathcal{O})) + \sum_{i=1}^{3}((P_i') - (\mathcal{O})) = 0$, which means that $D \sim D' = \sum_{i=1}^{3}((P_i') - (\mathcal{O}))$ in the

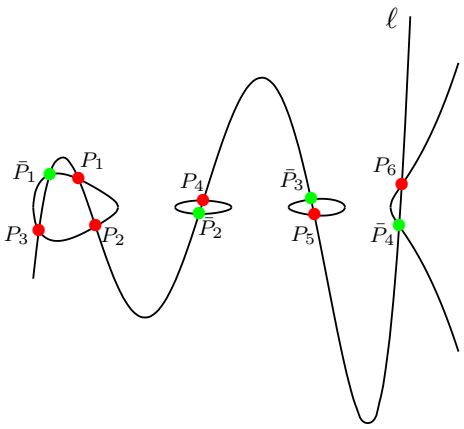


Figure 2.19: The first stage of reducing $D = \sum_{i=1}^{6}((P_i) - (\mathcal{O}))$.

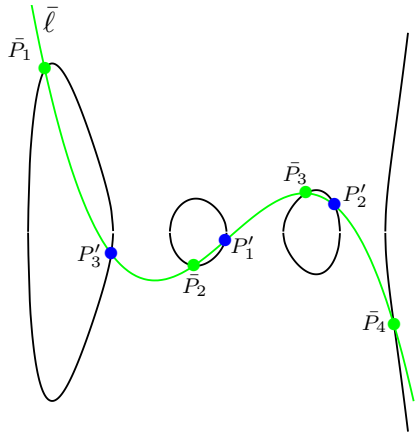


Figure 2.20: The second (and final) stage of divisor reduction.

divisor class group, and D' is the unique representative of D since $\mathrm{Deg}(\epsilon(D')) = 3 \leq g$.

We will not be discussing higher genus ($g \geq 2$) curves until Chapter 7, where among other things, we discuss the hyperelliptic analogy of the elliptic curve chord-and-tangent rule. The reason the bulk of this thesis works only on elliptic curves is because in the arena of pairing-based cryptography, the raw speed of elliptic curves is currently unrivalled by their higher genus counterparts, and all of the state-of-the-art implementations take place in the genus 1 setting.

The elliptic curve group law enjoys a (relatively speaking) very simple, almost entirely elementary description, the only exception being the introduction of projective space for the formal definition of $\mathcal{O}$. Namely, we were able to describe the chord-and-tangent rule without the language of divisors or the definition of the divisor class group, which is not the case for other curves or general abelian varieties. This is because of the one-to-one correspondence between the divisor class group $\mathrm{Pic}^0(E)$ and the points on E we briefly mentioned in Example 2.2.7, i.e. the group homomorphism $P \mapsto (P) - (\mathcal{O})$ (see [Sil09, III.3.4] [Gal12, Th. 7.9.8, Th. 7.9.9]). Thus, in the elliptic curve setting, we can simply talk about the group elements being points, rather than divisors. In higher genera this does not happen; group elements are no longer points, but rather divisor classes in $\mathrm{Pic}^0(E)$ with multiple elements in their support (again, more on this in Chapter

7).

Nevertheless, as we will see in the coming sections, the language of divisors is absolutely essential in the description of elliptic curve pairings, where the objective is to compute very large (degree) functions on E with prescribed divisors, and then evaluate these functions at other divisors[5]. Evaluating a function $f \in \mathbb{F}_q(E)$ at a divisor $D = \sum_{P \in E} n_P(P)$ has a natural definition, provided the divisors (f) and D have disjoint supports:

$$f(D) = \prod_{P \in E} f(P)^{n_P}. \tag{2.14}$$

The stipulation of disjoint supports is clearly necessary for $f(D)$ to be non-trivial, since $P \in \mathrm{supp}((f))$ implies P is a zero or pole of f on E, meaning $f(P)^{n_P}$ would be either zero or infinity respectively.

Example 2.2.11 (Magma script). Consider $E/\mathbb{F}_{163} : y^2 = x^3 - x - 2$, with $P = (43, 154)$, $Q = (46, 38)$, $R = (12, 35)$ and $S = (5, 66)$ all on E. Let $\ell_{P,Q}$, $\ell_{P,P}$ and $\ell_{Q,Q}$ be the lines joining P and Q, tangent to P, and tangent to Q on E respectively, computed as $\ell_{P,Q} : y+93x+85$, $\ell_{P,P} : y+127x+90$, $\ell_{Q,Q} : y+13x+16$. Let $D_1 = 2(R) + (S)$, $D_2 = 3(R) - 3(S)$ and $D_3 = (R) + (S) - 2(\mathcal{O})$. We can compute $\ell_{P,Q}(D_1) = (y_R + 93x_R + 85)^2(y_S + 93x_S + 85) = 122$, or $\ell_{P,P}(D_2) = (y_R + 127x_R + 90)^3/(y_S + 127x_S + 90)^3 = 53$, but we can not evaluate any of these functions at D_3, since $\mathcal{O} \in \mathrm{supp}(D_3)$, and $\mathcal{O}$ is also in the supports of $(\ell_{P,Q})$, $(\ell_{P,P})$, $(\ell_{Q,Q})$. Let $\ell'_{P,P} = 17\ell_{P,P}$ so that $\ell'_{P,P} = 17y + 40x + 63$, and that $\ell'_{P,P}(D_2) = (17y_R + 40x_R + 63)^3/(17y_S + 40x_S + 63)^3 = 53 = \ell_{P,P}(D_2)$. This is true in general, i.e. that if $g = cf$ for some constant $c \in \overline{\mathbb{F}}_q$, then $f(D) = g(D)$ if D has degree zero; the constant c will cancel out because $\mathrm{Deg}(D) = 0$ implies the numerator and denominator of $f(D)$ (identically $g(D)$) have the same total degree.

2.2.3 Weil reciprocity

We conclude our section on divisors (as Galbraith does [Gal05, §IX.2, Th. IX.3], where he also gives a proof) with a central theorem that lies at the heart of many of the proofs of cryptographic pairing properties.

[5] We will also see that we do not actually compute these very large functions explicitly before evaluating them.

Theorem 2.1 (Weil reciprocity)**.** *Let f and g be non-zero functions on a curve such that (f) and (g) have disjoint supports. Then $f((g)) = g((f))$.*

Most of the functions on E that we have seen so far contain $\mathcal{O}$ in their support. In the first example (2.2.12) we will choose one of the functions such that this is not the case, meaning that Theorem 2.1 can be applied instantly, whilst in the second example we will show how to alleviate this problem when it arises by modifying either of the functions.

Example 2.2.12 (Magma script). Let $E/\mathbb{F}_{503} : y^2 = x^3+1$. Consider the functions $f : \frac{20y+9x+179}{199y+187x+359} = 0$ and $g : y + 251x^2 + 129x + 201 = 0$ on E. The divisor of f is $(f) = 2(433, 98) + (232, 113) - (432, 27) - 2(127, 258)$, and the divisor of g is $(g) = (413, 369) + (339, 199) + (147, 443) + (124, 42) - 4(\mathcal{O})$. The supports are clearly disjoint, so we first compute $f((g))$ as

$$\frac{\left(\frac{20\cdot 369+9\cdot 413+179}{199\cdot 369+187\cdot 413+359}\right)\cdot\left(\frac{20\cdot 199+9\cdot 339+179}{199\cdot 199+187\cdot 339+359}\right)\cdot\left(\frac{20\cdot 443+9\cdot 147+179}{199\cdot 443+187\cdot 147+359}\right)\cdot\left(\frac{20\cdot 42+9\cdot 124+179}{199\cdot 42+187\cdot 124+359}\right)}{\left(\frac{20\cdot 1+9\cdot 0+179\cdot 0}{199\cdot 1+187\cdot 0+359\cdot 0}\right)^4} = 321.$$

Notice that f was cast into projective space as $f : \frac{20Y+9X+179Z}{199Y+187X+359Z}$ for the evaluation at $\mathcal{O} = (0 : 1 : 0)$ on the denominator. Now, for $g((f))$ we have

$$\frac{\left(98 + 251\cdot 433^2 + 129\cdot 433 + 201\right)^2\cdot\left(113 + 251\cdot 232^2 + 129\cdot 232 + 201\right)}{\left(258 + 251\cdot 127^2 + 129\cdot 127 + 201\right)^2\cdot\left(27 + 251\cdot 432^2 + 129\cdot 432 + 201\right)} = 321.$$

Example 2.2.13 (Magma script). Let $P, Q, R, S, T, U \in E$, such that $T = -(R + S)$. Further let $\ell' : y = (\lambda' x + \nu')$ be the tangent to E at P and $\ell : y = (\lambda x + \nu)$ be the line between R, S and T depicted in Figure 2.21, so that $(\ell') = 2(P) + (-[2]P) - 3(\mathcal{O})$ and $(\ell) = (R) + (S) + (T) - 3(\mathcal{O})$. Suppose we wish to compute $\ell(\ell')$.

At this point it does not make sense to compute $\ell(\ell')$ (or $\ell'(\ell)$) since $\mathrm{supp}((\ell)) \cap \mathrm{supp}((\ell')) = \{\mathcal{O}\}$. We can fix this by finding a divisor equivalent to, say (ℓ), whose support is disjoint to $\mathrm{supp}((\ell'))$. This is easily done by picking a random point $U \notin \mathrm{supp}(\ell')$ and defining $D = (R+U) + (S+U) + (T+U) - 3(U)$. To see that $D \sim \ell$, observe that $(R+U) - (U) = (R) - (\mathcal{O})$ by writing down the divisor of the quotient of the sloped and vertical lines in the addition of R and U on E. Computing $\ell(\ell')$ is therefore the same as computing $D(\ell')$, but this computation would then require finding a new function on E with divisor D, so we can invoke

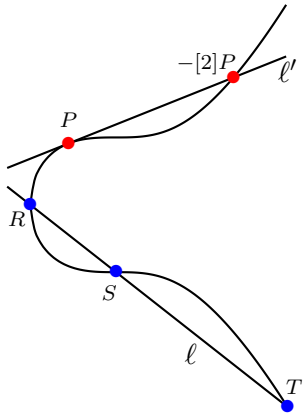


Figure 2.21: $\mathrm{supp}(\epsilon((\ell)))$ and $\mathrm{supp}(\epsilon((\ell')))$.

Theorem 2.1 and instead compute $\ell'(D)$ as

$$\ell'(D) = \frac{(y_{R'} - (\lambda' x_{R'} + \nu'))\,(y_{S'} - (\lambda' x_{S'} + \nu'))\,(y_{T'} - (\lambda' x_{T'} + \nu'))}{(y_U - (\lambda' x_U + \nu'))^3},$$

where $R' = (x_{R'}, y_{R'}) = R+U$, $S' = (x_{S'}, y_{S'}) = S+U$ and $T' = (x_{T'}, y_{T'}) = T+U$ are all such that $R', S', T' \notin \mathrm{Supp}(\ell')$, so that $\ell'(D)$ is the same as $\ell(\ell')$ by Weil reciprocity.

2.2.4 Section summary

We introduced the important concept of divisors on curves. We illustrated their particular usefulness when used to describe functions on curves, since such a function is well defined (up to constant) by its points of intersection with a curve, and these are precisely what the divisor of the function encapsulates. We defined the *divisor class group* of a (hyperelliptic) curve and discussed that for the case of elliptic curves, there is a bijection between this group and the set of points on the curve, so that we can simply talk about group elements as points on E rather than divisors. We further illustrated several useful properties and theorems that will be used throughout this thesis, but which also play a big role in the realm of algebraic geometry, most notably the Riemann-Roch theorem and Weil reciprocity. For the most part we specified the context to elliptic curves over finite fields, but all of the results and properties discussed above apply to arbitrary curves over arbitrary fields.

2.3 Elliptic curves as pairing groups

The purpose of this section is to define the elliptic groups that are used in cryptographic pairings. We start with the most abstract definition [Sil10]: a pairing is a *bilinear* map on an abelian group M taking values in some other abelian group R

$$\langle \cdot\ , \cdot \rangle : M \times M \to R.$$

Suppose that the binary group operations in M and R are respectively denoted by $+$ and $*$. The bilinearity property of the above map (that classifies it a pairing) means that, for $x, y, z \in M$, we have

$$\langle x + y, z \rangle = \langle x, z \rangle * \langle y, z \rangle,$$
$$\langle x, y + z \rangle = \langle x, y \rangle * \langle x, z \rangle.$$

That is, the map $\langle \cdot, \cdot \rangle$ is linear in both inputs.

It is this bilinearity property that makes pairings such a powerful primitive in cryptography. For our purposes we often find it advantageous to slightly relax the condition that the two arguments in the map come from the same group, and allow them to come from cyclic groups of the same order (which are therefore isomorphic). Thus, throughout this thesis and in the abundance of literature related to cryptography, the notation commonly used for the bilinear map is

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T.$$

Our primary objective in this section is to define the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The definition of $\mathbb{G}_T$ will come with the definition of the pairings in the next section.

Currently, the only known instantiations of pairings suitable for cryptography are the Weil and Tate pairings on divisor class groups of algebraic curves, and in the simplest and most efficient cases, on elliptic curves. Let $\mathbb{F}_{q^k}$ be some finite extension of $\mathbb{F}_q$ with $k \geq 1$. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are defined in $E(\mathbb{F}_{q^k})$, and the *target group* $\mathbb{G}_T$ is defined in the multiplicative group $\mathbb{F}^*_{q^k}$, so we usually write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively, whilst we write $\mathbb{G}_T$ multiplicatively. Thus, for $P, P' \in \mathbb{G}_1$

and $Q, Q' \in G_2$, the bilinearity of e means that

$$e(P + P', Q) = e(P, Q) \cdot e(P', Q),$$
$$e(P, Q + Q') = e(P, Q) \cdot e(P, Q'),$$

from which it follows that, for scalars $a, b \in \mathbb{Z}$, we have

$$e([a]P, [b]Q) = e(P, [b]Q)^a = e([a]P, Q)^b = e(P, Q)^{ab} = e([b]P, [a]Q). \quad (2.15)$$

Even though we are yet to define $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$, and we are still a while away from beginning the discussion of how the pairing $e(P, Q)$ is computed, it helps to immediately see the bilinearity property of pairings in context.

Example 2.3.1 (Magma script). Let $q = 7691$ and let $E/\mathbb{F}_q : y^2 = x^3+1$. Suppose $\mathbb{F}_{q^2}$ is constructed $\mathbb{F}_{q^2} = \mathbb{F}_q(u)$ where $u^2 + 1 = 0$. Let $P = (2693, 4312) \in E(\mathbb{F}_q)$ and $Q = (633u + 6145, 7372u + 109) \in E(\mathbb{F}_{q^2})$. $\#E(\mathbb{F}_q) = 2^2 \cdot 3 \cdot 641$ and $\#E(\mathbb{F}_{q^2}) = 2^4 \cdot 3^2 \cdot 641^2 = \#E(\mathbb{F}_q)^2$. P and Q were especially chosen (we will see why later) to be in different subgroups of the same prime order $r = |\langle P \rangle| = |\langle Q \rangle| = 641$. The Weil pairing $e(\cdot, \cdot)$ of P and Q is $e(P, Q) = 6744u + 5677 \in \mathbb{F}^*_{q^2}$. In fact, $r \mid \#\mathbb{F}_{q^2}$, and $e(P, Q)$ actually lies in a subgroup of $\mathbb{F}_{q^2}$, namely the r-th roots of unity $\mu_r \in \mathbb{F}_{q^2}$, meaning that $e(P, Q)^r = 1$. We are now in a position to illustrate some examples of bilinearity. Thus, take any $a \in \mathbb{Z}_r$ and $b \in \mathbb{Z}_r$, say $a = 403$ and $b = 135$, and see that $[a]P = (4903, 2231)$ and $[b]Q = (5806u+1403, 6091u+2370)$. We can compute $e([a]P, Q) = 3821u+7025$ and verify that $e([a]P, Q) = 3821u + 7025 = (6744u + 5677)^{403} = e(P, Q)^a$; or $e(P, [b]Q) = 248u + 5$ to see that $e(P, [b]Q) = 248u + 5 = (6744u + 5677)^{135} = e(P, Q)^b$; or $e([a]P, [b]Q) = 2719u + 2731 = (6744u + 5677)^{561} = e(P, Q)^{a \cdot b \bmod r}$. Note that since $e(P, Q) \neq 1 \in \mu_r$, $e([a]P, [b]Q)$ will only be trivial if $r \mid ab$, which implies $r \mid a$ or $r \mid b$, meaning either (or both) of $[a]P$ or $[b]Q$ must be $\mathcal{O}$. Thus, $e(P, Q) \neq 1$ guarantees non-trivial pairings for non-trivial arguments; this is a cryptographically necessary property that is called *non-degeneracy*.

Following Example 2.3.1 above, if a pairing e is bilinear, non-degenerate and efficiently computable, e is called an *admissible pairing*.

Remark 2.3.1 (ECC vs. PBC). This informal remark is intended as a point of clarification for PBC newcomers. Our confusion in the early days of digesting the vast amount of literature was in part alleviated by one paragraph in Lynn's thesis that helped put the relationship between ECC and PBC in a wider context. The

only known admissible pairings that are suitable for cryptography are the Weil and Tate pairings on algebraic curves. The fact that these pairings can be defined on elliptic curves, which were already a highly attractive cryptographic setting before pairings arrived on the scene, is, as Lynn puts it, a "happy coincidence". Cryptographers would have welcomed secure, admissible pairings in any suitable form, but the fact that they were handed down from the realm of algebraic geometry and are computed on elliptic curves makes them "even more attractive" [Lyn07, §2.9].

In cryptography we need more properties than the three which constitute an admissible pairing. The magic of the bilinearity property in (2.15) that gives pairing-based primitives increased functionality over traditional primitives is useless unless discrete logarithm related problems within all three groups remain intractable. Example 2.3.1 gives an admissible pairing, but because the toy sizes of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ clearly offer no resistance in regards to their respective discrete logarithm problems, such a pairing instance would clearly never be used. However, if the size r of all three groups was inflated to be much larger (say 512 bits), then the corresponding pairing could meet current security requirements and resist all known attacks. We present an alternative bilinear pairing that meets the admissible requirements, but (regardless of how large the group sizes are) is still not suitable for PBC. This example too, is taken from Lynn's thesis [Lyn07, §1.9].

Example 2.3.2 (Magma script). Let $r > 1$ be an integer. Suppose $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ has $\mathbb{G}_1 = \mathbb{G}_T = \mathbb{Z}_r^*$ and $\mathbb{G}_2 = \mathbb{Z}_{r-1}^+$, and is defined by $e : (g, a) = g^a$. Notice that for $g, g' \in \mathbb{G}_1$, we have $e(g \cdot g', a) = e(g, a) \cdot e(g', a)$, and for $a, a' \in \mathbb{G}_2$ we have $e(g, a + a') = e(g, a) \cdot e(g, a')$. Although e is then clearly bilinear, the discrete logarithm problem in $\mathbb{G}_2$ is easy, so the power of the bilinear map becomes somewhat redundant. It is interesting to see, however, that we can still state some of the classical problems in terms of the above pairing. For example, if we set r to be a large prime, then the standard discrete logarithm problem becomes: given $g \in \mathbb{G}_1$, $h \in \mathbb{G}_T$, find $a \in \mathbb{G}_2$ such that $e(g, a) = h$.

2.3.1 The r-torsion

We now turn our focus towards concretely defining the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Having not yet seen how pairings are computed, we will need to make some statements regarding what we need out of $\mathbb{G}_1$ and $\mathbb{G}_2$ that will really only tie together when

the definitions of the Weil and Tate pairings come in the following section. The main such statement is that computing the pairing $e(P, Q)$, in either the Weil or Tate sense, requires that P and Q come from disjoint cyclic subgroups of the same prime[6] order r. At this point we can only hint towards why by referring back to the stipulation of disjoint supports that was made in the statement of Weil reciprocity (Theorem 2.1), and claiming that if P and Q are in the same cyclic subgroup, then the pairing computation essentially fails because supports of the associated divisors are forced to undesirably coincide.

We have already seen an example (2.3.1) of how we can find more than one cyclic subgroup of order r, when $E(\mathbb{F}_q)$ itself only contains one subgroup. Namely, we extended $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ and saw that $E(\mathbb{F}_{q^2}) \setminus E(\mathbb{F}_q)$ had at least one other subgroup of order r, where we were able to define Q and subsequently compute $e(P, Q)$. This is precisely the way we obtain two distinct order-r subgroups in general: we find the smallest extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$ such that $E(\mathbb{F}_{q^k})$ captures more points of order r. The integer $k \geq 1$ that achieves this is called the *embedding degree*, and it plays a crucial role in pairing computation. Also at the heart of our discussion then, is the entire group of points of order r on $E(\overline{\mathbb{F}}_q)$, called the r-torsion, which is denoted by $E[r]$ and defined as $E[r] = \{P \in E : [r]P = \mathcal{O}\}$. The following result (see [ACD+05, Th. 13.13] or [Sil09, Ch. III, Cor. 6.4(b)]) is quite remarkable; it tells us not only the cardinality of $E[r]$, but its structure too. If K is any field with characteristic zero or prime to r, we have

$$E[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r. \tag{2.16}$$

This means that in general, $\#E[r] = r^2$. Furthermore, since the point at infinity $\mathcal{O}$ overlaps into all order r subgroups, Equation (2.16) implies that (for prime r) the r-torsion consists of $r+1$ cyclic subgroups of order r. The following equivalent conditions for the embedding degree k also tell us precisely where $E[r]$ lies in its entirety. We note that the embedding degree is actually a function $k(q, r)$ of q and r, but we just write k since the context is usually clear.

- k is the smallest positive integer such that $r \mid (q^k - 1)$;
- k is the smallest positive integer such that $\mathbb{F}_{q^k}$ contains all of the r-th roots of unity in $\overline{\mathbb{F}}_q$ (i.e. $\mu_r \subset \mathbb{F}_{q^k}$);

[6] There has been some work that exploits additional functionality if r is composite, e.g. an RSA modulus $n = pq$, but we do not consider this much less common and much less efficient setting – see [BGN05, Fre10, BRS11, Lew12] for more details.

- k is the smallest positive integer such that $E[r] \subset E(\mathbb{F}_{q^k})$.

If $r \| \#E(\mathbb{F}_q)$ (i.e. $r \mid \#E(\mathbb{F}_q)$ but $r^2 \nmid \#E(\mathbb{F}_q)$), then the r-torsion subgroup in $E(\mathbb{F}_q)$ is unique. In this case, $k > 1$ and (2.16) implies that $\mathbb{F}_{q^k}$ is the smallest field extension of $\mathbb{F}_q$ which produces any more r-torsion points belonging to $E(\mathbb{F}_{q^k}) \setminus E(\mathbb{F}_q)$. In other words, once the extension field is big enough to find one more point of order r (that is not defined over the base field), then we actually find all of the points in $E[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r$. Scott [Sco04] describes this phenomenon more poetically:

> "... something rather magical happens when a curve with the same equation is considered over the field $\mathbb{F}_{q^k}$ for a certain value of k. The group structure undergoes a strange blossoming, and takes on a new, more exotic character."

We also find Scott's depiction of the torsion subgroup $E[r]$ especially instructive [Sco04, Sco07a], so we use it in the following examples and throughout the rest of this section.

Example 2.3.3 (Magma script). Let $q = 11$, and consider $E/\mathbb{F}_q : y^2 = x^3 + 4$. $E(\mathbb{F}_q)$ has 12 points, so take $r = 3$ and note (from Equation (2.16)) that there are 9 points in the 3-torsion. Only 3 of them are found in $E(\mathbb{F}_q)$, namely $(0, 2)$, $(0, 9)$ and $\mathcal{O}$, which agrees with the fact that the embedding degree $k \neq 1$, since $(q^1 - 1) \not\equiv 0 \bmod r$. However, $(q^2 - 1) \equiv 0 \bmod r$ which means that the embedding degree is $k = 2$, so we form $\mathbb{F}_{q^2} = \mathbb{F}_q(u)$, with $u^2 + 1$. Thus, we are guaranteed to find the whole 3-torsion in $E(\mathbb{F}_{q^2})$, and it is structured as 4 cyclic subgroups of order 3; $\mathcal{O}$ overlaps into all of them – see Figure 2.22. We point out that although $\mathcal{O}$ is in the 3-torsion, it does not have order 3, but rather order 1 – points of order $d \mid r$ are automatically included in the r-torsion. Take any two points $P, Q \in E[3] \setminus \{\mathcal{O}\}$ that are not in the same subgroup, neither of which are $\mathcal{O}$. The translation of Equation (2.16) is that any other point in $E[3]$ can be obtained as $[i]P + [j]Q$, $i, j \in \{0, 1, 2\}$. Fixing $P \neq \mathcal{O}$ and letting j run through $0, 1, 2$ lands $P + [j]Q$ in the other three subgroups of $E[3]$ (that are not $\langle Q \rangle$ – this corresponds to $P = \mathcal{O}$).

Example 2.3.4 (Magma script). In the rare case that $r^2 \mid \#E$, it is possible that the entire r-torsion can be found over $E(\mathbb{F}_q)$, i.e. that the embedding degree is $k = 1$. Consider $E/\mathbb{F}_{31} : y^2 = x^3 + 13$, which has 25 points, so take $r = 5$. Since $r \mid q - 1$, $k = 1$ and therefore $E[r] \subseteq E(\mathbb{F}_q)$; Figure 2.23 show the 6 cyclic

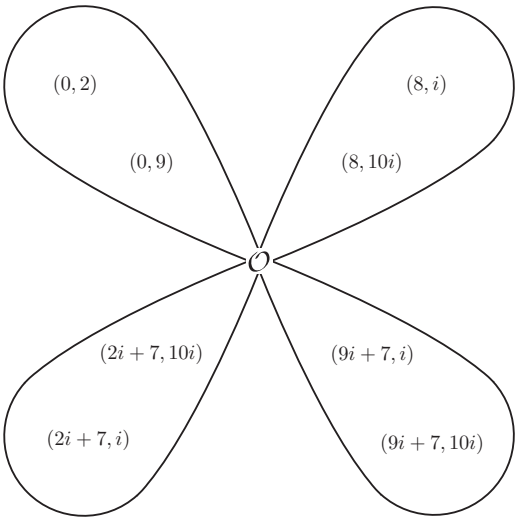


Figure 2.22: The 3-torsion: $E[3]$.

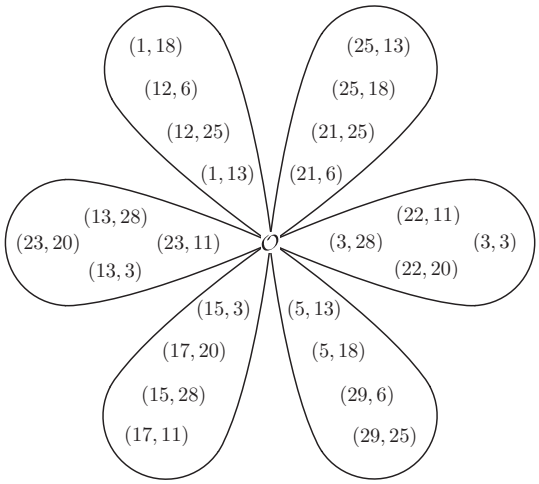


Figure 2.23: The 5-torsion: $E[5]$.

subgroups of order 5 constituting $E[5] \cong \mathbb{Z}_5 \times \mathbb{Z}_5$. Of course, $r^2 \mid \#E(\mathbb{F}_q)$ does not necessarily imply that $E[r] \subseteq E(\mathbb{F}_q)$, as points of order r^2 are possible.

Before the next example, we introduce an important map that plays an intricate role within the r-torsion subgroups. Since we are working over finite extension fields of $\mathbb{F}_q$, it is natural that we find a useful contribution from Galois

theory. Namely, the *trace map* of the point $P = (x, y) \in E(\mathbb{F}_{q^k})$ is defined as

$$\mathrm{Tr}(P) = \sum_{\sigma \in \mathrm{Gal}(\mathbb{F}_{q^k}/\mathbb{F}_q)} \sigma(P) = \sum_{i=0}^{k-1} \pi^i(P) = \sum_{i=0}^{k-1} (x^{q^i}, y^{q^i}),$$

where π is the q-power Frobenius endomorphism defined in Equation (2.7). Galois theory tells us that $\mathrm{Tr} : E(\mathbb{F}_{q^k}) \to E(\mathbb{F}_q)$, so when $r \| \#E(\mathbb{F}_q)$ (which will always be the case from now on), then this map, which is actually a group homomorphism, sends all torsion points into one subgroup of the r-torsion. We illustrate in Example 2.3.5 before painting the general picture.

Example 2.3.5 (Magma script). We take $q = 11$ again, but this time with $E/\mathbb{F}_q : y^2 = x^3+7x+2$. $E(\mathbb{F}_q)$ has 7 points, so take $r = 7$. We already have $E(\mathbb{F}_q)[r]$, but to collect $E[r]$ in its entirety we need to extend $\mathbb{F}_q$ to $\mathbb{F}_{q^k}$. This time, the smallest integer k such that $(q^k - 1) \bmod 7 \equiv 0$ is $k = 3$, so we form $\mathbb{F}_{q^3} = \mathbb{F}_q(u)$ with $u^3+u+4 = 0$, and we are guaranteed that $E[7] \subset E(\mathbb{F}_{q^3})$. The entire 7-torsion has

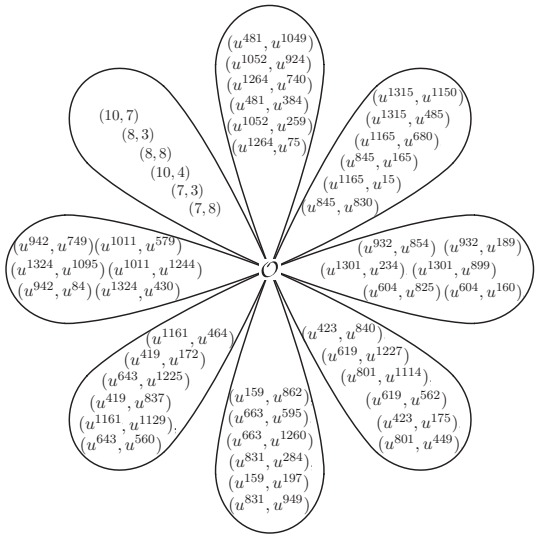


Figure 2.24: The 7-torsion: $E[7]$.

cardinality 49 and splits into 8 cyclic subgroups, as shown in Figure 2.23. To fit the points in, we use the power representation of elements in $\mathbb{F}_{q^3} = \mathbb{F}_q(u)$. In this case, for $P \in E(\mathbb{F}_{q^3})$, the trace map on E is $\mathrm{Tr}(P) = (x, y) + (x^q, y^q) + (x^{q^2}, y^{q^2})$. For the unique torsion subgroup $E(\mathbb{F}_q)[r]$, the Frobenius endomorphism is trivial ($\pi(P) = P$) so the trace map clearly acts as multiplication by k, i.e. $\mathrm{Tr}(P) = [k]P$. However, Tr will send every other element in the torsion into $E(\mathbb{F}_q)[r]$.

For example, for $Q = (u^{481}, u^{1049})$ (in the subgroup pointing upwards), we have $\mathrm{Tr}(Q) = (8, 8)$; for $R = (u^{423}, u^{840})$ (the lower right subgroup), we have $\mathrm{Tr}(R) = (10, 7)$; for $S = (u^{1011}, u^{1244})$, we have $\mathrm{Tr}(S) = (8, 3)$. There is one other peculiar subgroup in $E[7]$ however, for which the trace map sends each element to $\mathcal{O}$. This occurs in general, and we are about to see that this has important consequences in PBC, but in our case this subgroup is the upper right group containing $T = (u^{1315}, u^{1150})$, i.e. $\mathrm{Tr}(T) = \mathcal{O}$, so $\mathrm{Tr} : \langle T \rangle \to \{\mathcal{O}\}$. One final point to note is that the embedding degree $k = 3$ also implies that the (six) non-trivial 7-th roots of unity are all found in $\mathbb{F}_{q^3}$ (but not before), i.e. $\mu_7 \setminus \{1\} \in \mathbb{F}_{q^3} \setminus \mathbb{F}_{q^2}$.

We now give a general depiction of the r-torsion $E[r]$. To do so, we need to discuss a few assumptions that apply most commonly to the scenarios we will be encountering. Firstly, we assume that $r \| \#E(\mathbb{F}_q)$ is prime and the embedding degree k (with respect to r) is $k > 1$. Thus, there is a unique subgroup of order r in $E[r]$ which is defined over $\mathbb{F}_q$, called the *base-field* subgroup; it is denoted by $\mathcal{G}_1$. Since the Frobenius endomorphism π acts trivially on $\mathcal{G}_1$, but nowhere else in $E[r]$, then it can be defined as $\mathcal{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$. That is, $\mathcal{G}_1$ is the $[1]$-*eigenspace* of π restricted to $E[r]$. There is another subgroup of $E[r]$ that can be expressed using an eigenspace of π. Referring back to Equation (2.8), we can easily deduce that the other eigenvalue of π is q, and we define another subgroup $\mathcal{G}_2$ of $E[r]$ as $\mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$. It turns out that this subgroup is precisely the peculiar subgroup we alluded to in Example 2.3.5. We call $\mathcal{G}_2$ the *trace zero* subgroup, since all $P \in \mathcal{G}_2$ have $\mathrm{Tr}(P) = \mathcal{O}$; this result is attributed to Dan Boneh [Gal05, Lemma IX.16]. We illustrate in Figure 2.25.

We can also map any $P \in E[r]$ to the trace zero subgroup $\mathcal{G}_2$ via the *anti-trace map* $\mathrm{aTr} : P \mapsto P' = [k]P - \mathrm{Tr}(P)$; showing that $\mathrm{Tr}(P') = \mathcal{O}$ is a worthwhile exercise for the reader.

To define our pairing, we need to specify the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$: these $\mathbb{G}$'s are not to be confused with the $\mathcal{G}$'s that stand for two specific r-torsion subgroups, as $\mathbb{G}_1$ and $\mathbb{G}_2$ can be defined as any of the $r + 1$ groups in $E[r]$. As we will see however, there are many reasons we would like to specifically set $\mathbb{G}_1 = \mathcal{G}_1$ and $\mathbb{G}_2 = \mathcal{G}_2$, but as we will also see there are reasons that we may not want this to be the case. The existence of maps to and from the different torsion subgroups affects certain functionalities that cryptographers desire in a pairing-based protocol. These functionalities and the choices that are available to us will be discussed in a moment, but we must first look at one last map that

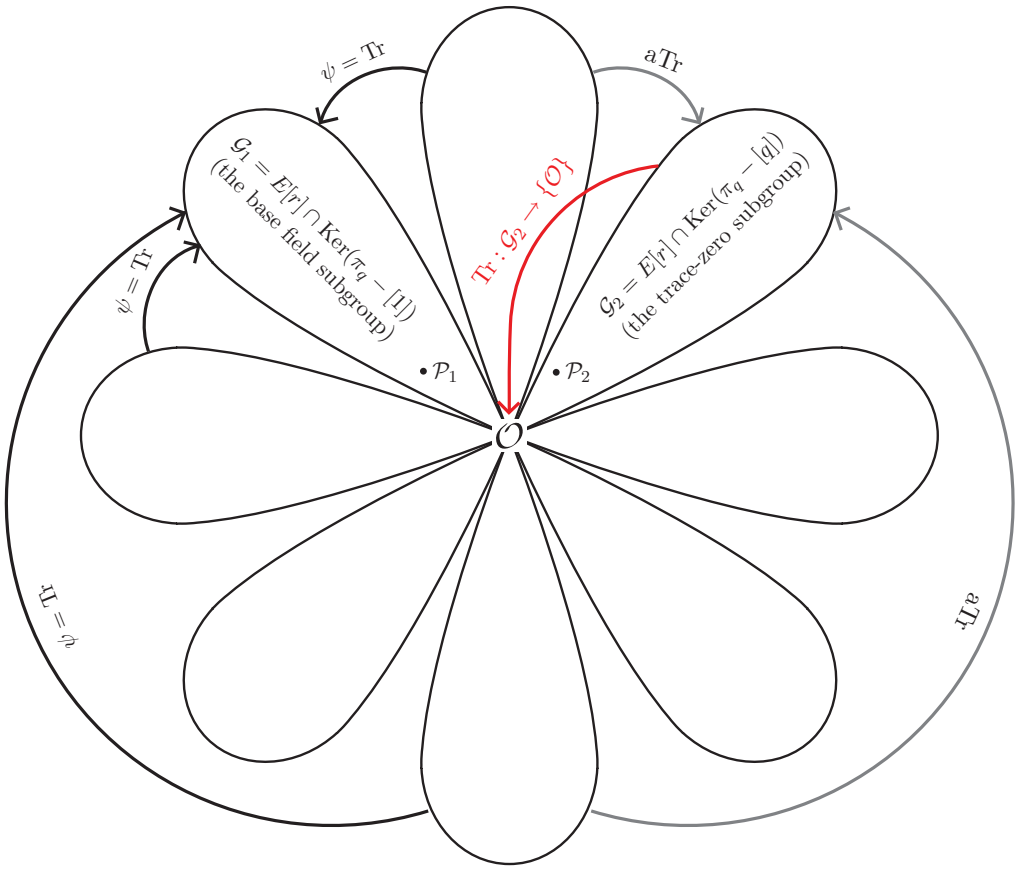


Figure 2.25: The behaviour of the trace and anti-trace maps on $E[r]$.

is available for a special class of curves.

Over prime fields, we call an elliptic curve E *supersingular*[7] if $\#E(\mathbb{F}_q) = q+1$. There are several other equivalent conditions [Sil09, Ch. V, Th. 3.1(a)], but the most meaningful property for our purposes is that a supersingular curve comes equipped with a *distortion map* ϕ; this is a non-($\mathbb{F}_q$-)rational map that takes a point in $E(\mathbb{F}_q)$ to a point in $E(\mathbb{F}_{q^k})$ [Gal05, §IX.7.2]. A curve which is not supersingular is called an *ordinary* curve, and it does not have such a map [Ver01, Th. 11]. We give two examples of elliptic curves that are supersingular, and show the behaviour of the distortion map ϕ within the torsion.

Example 2.3.6 (Magma script). Let $q = 59$, for which $E/\mathbb{F}_q : y^2 = x^3 + 1$ is supersingular, meaning $\#E(\mathbb{F}_q) = q + 1 = 60$, so take $r = 5$. The embedding

[7] This terminology should not be confused with the *singular* vs. *non-singular* definitions illustrated in, and discussed above, Figures 2.1-2.4.

degree is $k = 2$, so we construct the extension as $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$, $i^2 + 1 = 0$. $\xi_3 = 24i + 29$ is a cube root of unity, for which the associated distortion map is $\phi : (x, y) \mapsto (\xi_3 x, y)$. The fact that ϕ^3 is equivalent to the identity map on E is illustrated in Figure 2.26.

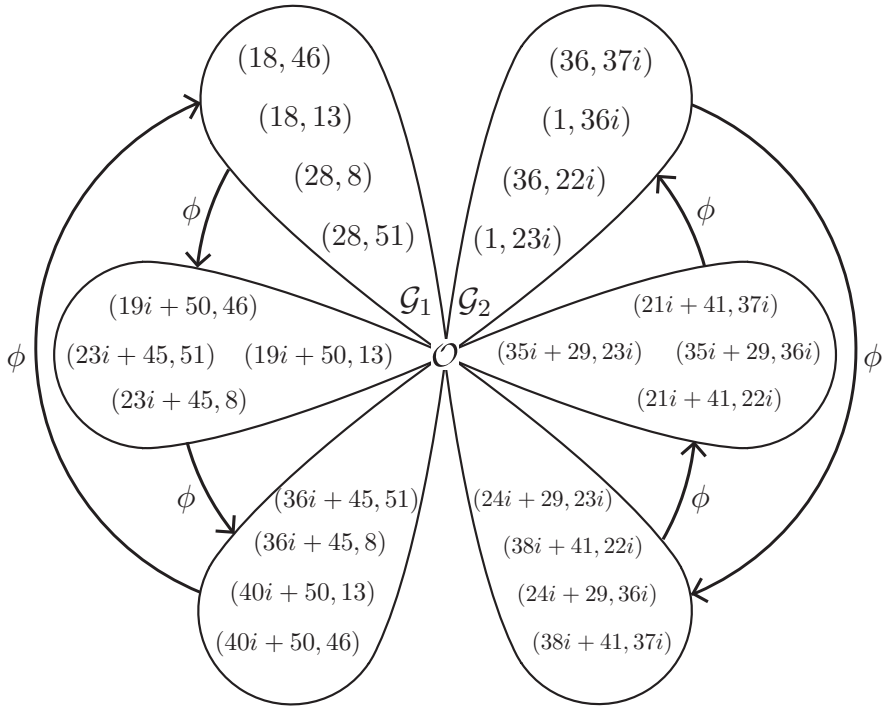


Figure 2.26: The distortion map $\phi : (x, y) \mapsto (\xi_3, y)$ on $E[5]$.

Example 2.3.7 (Magma script). We take the same fields as the last example ($q = 59$, $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$, $i^2 + 1 = 0$), but instead use the supersingular curve $E/\mathbb{F}_q : y^2 = x^3 + x$, which therefore also has $\#E(\mathbb{F}_q) = 60$. This time, the distortion map is $\phi : (x, y) \mapsto (-x, iy)$, from which it is easy to see that ϕ^4 is equivalent to the identity map on E. In Figure 2.27, we see that (in this case) the distortion map does not always move elements out of their subgroup, but rather restricting ϕ to, say the torsion subgroup generated by $(28i+51, 25i+49)$, gives an endomorphism on $\langle(28i + 51, 25i + 49)\rangle$. This hints towards one of the major optimisations in pairing computations. Namely, in Section 2.1 we saw the power of endomorphisms applied to ECC (specifically in Example 2.1.23), and in Section 2.6 we are going to see that endomorphisms on torsion subgroups (like the one above) can be used to great effect in PBC.

We summarise the available maps within the r-torsion. From any subgroup

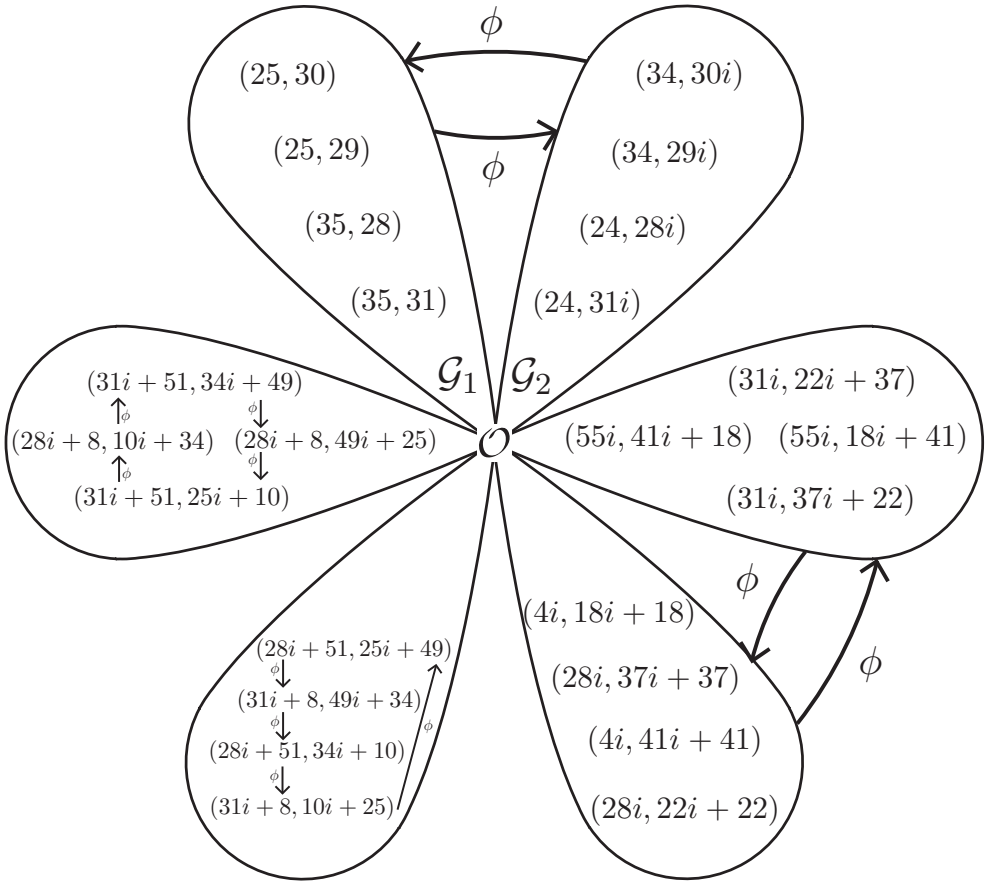


Figure 2.27: The distortion map $\phi : (x, y) \mapsto (-x, iy)$ on $E[5]$.

in $E[r]$ that is not $\mathcal{G}_1$ or $\mathcal{G}_2$, we can always map into either $\mathcal{G}_1$ or $\mathcal{G}_2$ via the trace and anti-trace maps respectively. If E is ordinary, we do not have computable maps out of $\mathcal{G}_1$ or $\mathcal{G}_2$, otherwise if E is supersingular then the distortion map ϕ is a homomorphic map out of these two subgroups.

2.3.2 Pairing types

As we mentioned before the previous two examples, the interplay between the maps that are available in any given scenario gives rise to different functionalities within a pairing-based protocol. Galbraith *et al.* [GPS08] were the first to identify that all of the potentially desirable properties in a protocol cannot be achieved simultaneously, and therefore classified pairings into certain *types*. There are now four pairing types in the literature; Galbraith *et al.* originally presented three, but a fourth type was added soon after by Shacham [Sha05]. The pairing types

essentially arise from observing the (practical) implications of placing $\mathbb{G}_1$ and $\mathbb{G}_2$ in different subgroups of $E[r]$; in fact, it will soon become obvious that it is always best to set $\mathbb{G}_1 = \mathcal{G}_1$, so the four types really are tied to the definition of $\mathbb{G}_2$. The main factors affecting the classification are the ability to hash and/or randomly sample elements of $\mathbb{G}_2$, the existence of an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ which is often required to make security proofs work (see [GPS08]), and (as always) issues concerning storage and efficiency.

We follow the notation and descriptions of Chen *et al.* [CCS07], and describe each pairing type in turn. The illustrations of each type are in Figures 2.28-2.31, where the base-field group $\mathcal{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ with generator $\mathcal{P}_1$ is always in the top left, whilst the trace-zero subgroup $\mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$ with generator $\mathcal{P}_2$ is always in the top right. Let P_1 be the generator of $\mathbb{G}_1$ and P_2 be the generator of $\mathbb{G}_2$. It should be born in mind that the pairing $e(P, Q)$ will only compute non-trivially if P and Q are in different subgroups.

- *Type 1 pairings.* This is the scenario where E is supersingular, meaning we can map out of $\mathcal{G}_1$ with ϕ. Thus, we set $\mathbb{G}_1 = \mathbb{G}_2 = \mathcal{G}_1$ (with $P_1 = P_2 = \mathcal{P}_1$). When it comes time to compute a pairing e between say P and Q, we can use ϕ to map Q to $\phi(Q)$ and define $e(P, Q) = \hat{e}(P, \phi(Q))$, where $\hat{e}$ is the Weil or Tate pairing. There are no hashing problems (getting into $E(\mathbb{F}_q)[r]$ requires a simple cofactor multiplication once we have hashed into $E(\mathbb{F}_q)$) and we trivially have an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$. The drawback of Type 1 pairings comes when considering bandwidth and efficiency: as we will see in Section 2.5, the condition that E be supersingular is highly restrictive when it comes to optimising the speed of computing the pairing. See Figure 2.28.

The remaining three cases are defined over ordinary elliptic curves, so (as we will again see in Section 2.5) there are no restrictions imposed on the choice of elliptic curve that lead to a loss of efficiency. For all these situations we have $\mathbb{G}_1 = \mathcal{G}_1$ and $P_1 = \mathcal{P}_1$ (where hashing is relatively easy), so we only need to discuss the choices for $\mathbb{G}_2$ and P_2.

- *Type 2 pairings.* In this situation we take $\mathbb{G}_2$ to be any of the $r-1$ subgroups in $E[r]$ that is not $\mathcal{G}_1$ or $\mathcal{G}_2$. We have the map $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ as the trace map Tr. We can also use the anti-trace map to move elements from $\mathbb{G}_2$ into $\mathcal{G}_2$ for efficiency purposes. The drawback is that there is no known

way of hashing into $\mathbb{G}_2$ specifically, or to generate random elements of $\mathbb{G}_2$. The best we can do here is to specify a generator $P_2 \in \mathbb{G}_2$ and generate elements via scalar multiplications of P_2, but this is often undesirable in protocols since we cannot generate random elements without knowing the discrete logarithm with respect to P_2. See Figure 2.29.

- *Type 3 pairings.* In this scenario we take $\mathbb{G}_2 = \mathcal{G}_2$, the trace zero subgroup. We can now hash into $\mathbb{G}_2$, at the very least by following a cofactor multiplication in $E(\mathbb{F}_{q^k})$ by the anti-trace map $\mathrm{aTr} : E[r] \to \mathcal{G}_2$ (we will soon see that there is a much more efficient way than this). The ironic drawback here is that the only subgroup (besides $\mathcal{G}_1$) that we can hash into is also the only subgroup we can not find a map out of. An isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ trivially exists, we just do not have an efficient way to compute it. Thus, security proofs that rely on the existence of such a ψ are no longer applicable, unless the underlying problem(s) remains hard when the adversary is allowed oracle access to ψ [SV07]. See Figure 2.30.

- *Type 4 pairings.* In this situation we take $\mathbb{G}_2$ to be the whole r-torsion $E[r]$, which is a group of order r^2. Hashing into $\mathbb{G}_2$ is possible, but not very efficient, however we cannot hash into the particular subgroup generated by any specific P_2 (i.e. $\mathbb{G}_2$ is not cyclic). Note that hashing into $E[r]$ will only give an element in $\mathcal{G}_1$ or $\mathcal{G}_2$ (which is undesirable in this case) with negligibly low probability for large r. See Figure 2.31.

Prior to these different situations being brought to the attention of the PBC community [GPS08], authors publishing pairing-based protocols were often incorrectly assuming combinations of the associated properties that could not be achieved in practice. The message to designers of pairing-based protocols was that individual attention is required to prescribe the pairing type which best suits any particular pairing instantiation. Whilst some authors have since followed this advice closely, a good example being [CCS07, Tables 1-6], it still seems most common that designers of pairing protocols take the easy way out and assume a Type 1 pairing. This approach is somewhat justified, as it allows cryptographers to avoid getting bogged down in the complex details of pairings whilst still enjoying all their functional properties, but overall it is less than satisfactory. The reason is that, at current levels of security, a Type 1 pairing is orders of magnitude more costly than say, a Type 3 pairing. Nowadays all of

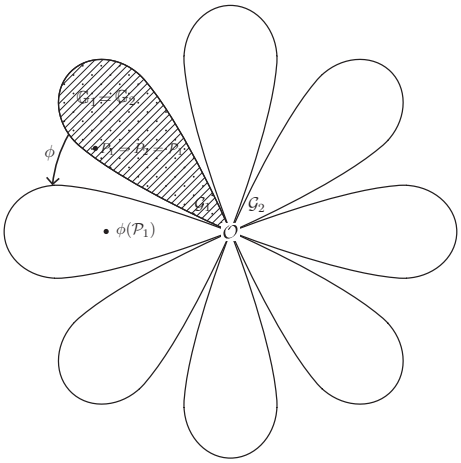


Figure 2.28: Type 1 pairings.

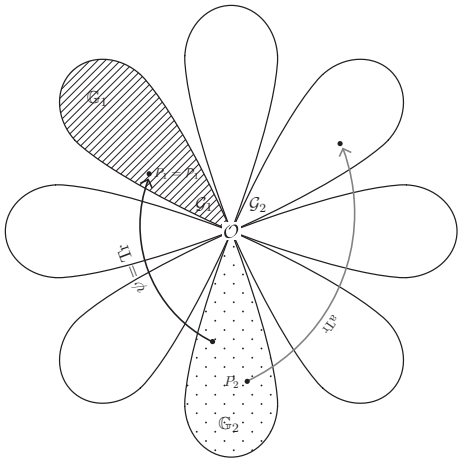


Figure 2.29: Type 2 pairings.

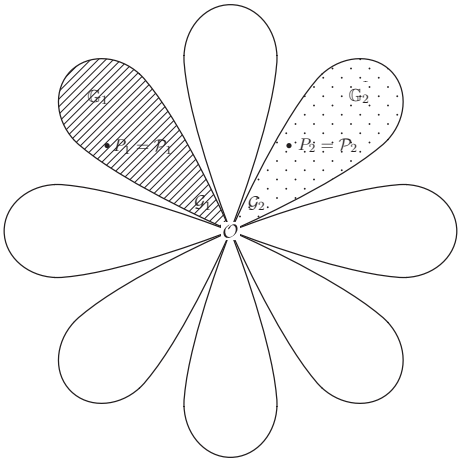


Figure 2.30: Type 3 pairings.

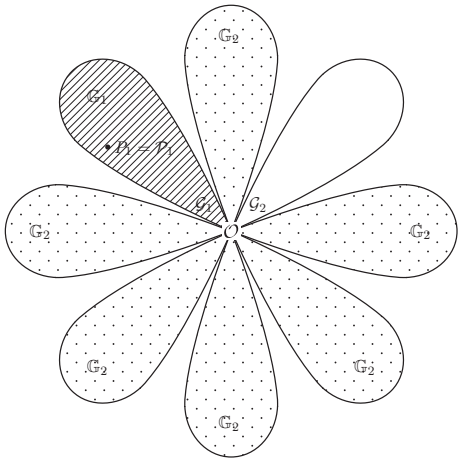


Figure 2.31: Type 4 pairings.

the state-of-the-art implementations of pairings take place on ordinary curves that assume the Type 3 scenario, where the only potential[8] sacrifice is the map $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$. Moreover, Chatterjee and Menezes [CM09] paid closer attention to the role of ψ in protocol (proof) designs and essentially argue that there is no known protocol/proof of security that cannot be translated into the Type 3 setting, claiming that Type 2 pairings (which are less efficient but have ψ) are merely inefficient implementations of Type 3 pairings. We note that their claim is only based on empirical evidence; they posed a counter-example as an open problem. Nevertheless, the final message of Menezes' related ECC2009 talk is

[8]The are some protocols whose security actually relies on the inability to compute ψ efficiently.

that "protocol designers who are interested in the performance of their protocols should describe and analyse their protocols using Type 3 pairings" [Men09].

For the remainder of this thesis then, and unless otherwise stated, the reader should assume we are in the Type 3 scenario where $\mathbb{G}_1 = \mathcal{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ and $\mathbb{G}_2 = \mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$.

2.3.3 Twisted curves

Before moving our focus to the algorithm for computing pairings, we have one final point to discuss; namely, the most efficient way to hash to, and represent elements in $\mathbb{G}_2$. This discussion brings up the crucial notion of *twists* of elliptic curves, which was first applied to pairings by Barreto *et al.* [BLS03]. We start with an example.

Example 2.3.8 (Magma script). Recall the curve used in Example 2.3.3: $q = 11$, $E/\mathbb{F}_q : y^2 = x^3 + 4$, $\#E(\mathbb{F}_q) = 12$ and $r = 3$. Excluding $\mathcal{O}$, the trace zero subgroup $\mathcal{G}_2$ consists of points defined in $E(\mathbb{F}_{q^2})$, namely $(8, i)$ and $(8, 10i)$. Define the curve $E'/\mathbb{F}_q : y^2 = x^3 - 4$ and observe that the map Ψ^{-1} defined by $\Psi^{-1} : (x, y) \mapsto (-x, iy)$ takes points from E to E', i.e. $\Psi^{-1} : E \to E'$. Restricting Ψ^{-1} to $\mathcal{G}_2$ actually gives a map that takes elements defined over $\mathbb{F}_{q^2}$ to elements defined over $\mathbb{F}_q$: $\Psi^{-1}((8, i)) = (3, 10)$ and $\Psi^{-1}((8, 10i)) = (3, 1)$. The convention is to write Ψ for the reverse map $\Psi : E' \to E$ which in this case is defined by $\Psi : (x', y') \mapsto (-x', y'/i) = (-x', -y'i)$. We call E' a *twist* of E. Every twist has a degree d, which tells us the extension field of $\mathbb{F}_q$ where E and E' become isomorphic. For our purposes, d is also the degree of its field of definition of E' as a subfield of $\mathbb{F}_{q^k}$, i.e. a degree d twist E' of E will be defined over $\mathbb{F}_{q^{k/d}}$. In this example, $k = 2$ and E' is defined over $\mathbb{F}_q$, so we are using a $d = 2$ twist, called a *quadratic twist*. Ordinarily, computations in the group $\mathbb{G}_2 = \mathcal{G}_2$ would require (point doubling/addition) operations in the extension field $\mathbb{F}_{q^2}$, but we can use Ψ^{-1} to instead perform these operations in $E'(\mathbb{F}_q)$, before mapping the result back with Ψ. Moreover, if we restrict the maps to $E[r]$, then Ψ^{-1} takes elements of the trace zero subgroup $\mathcal{G}_2$ of E and moves them to the base field subgroup $\mathcal{G}'_1$ of E'. Note that computing Ψ and Ψ^{-1} is essentially cost free.

We give a larger example that better illustrates the power of employing twisted curves.

Example 2.3.9 (Magma script). Let $q = 103$ and consider $E/\mathbb{F}_q : y^2 = x^3 + 72$, which has $\#E(\mathbb{F}_q) = 84$, so let $r = 7$. The embedding degree (with respect to r) is

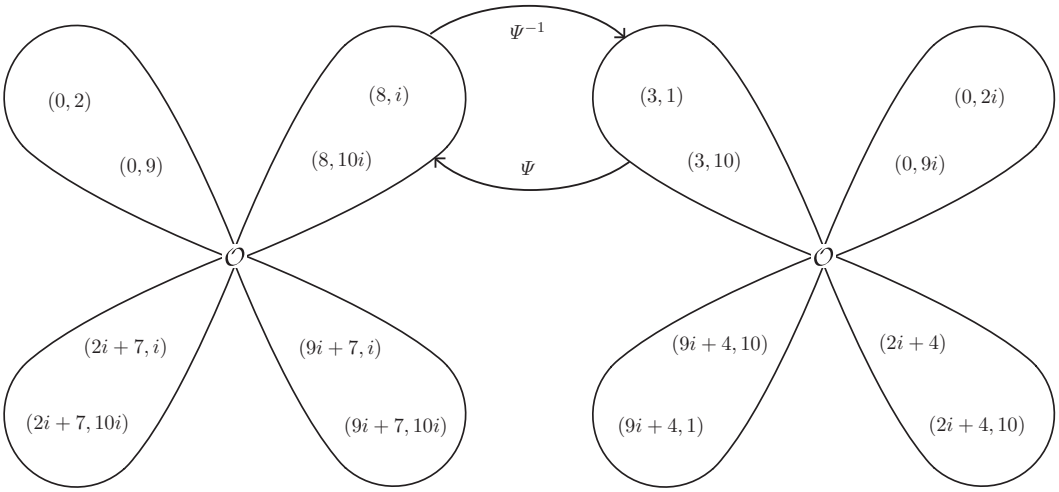


Figure 2.32: E (left) and the quadratic twist E' (right).

$k = 6$, so form $\mathbb{F}_{q^6} = \mathbb{F}_q(u)$ with $u^6 + 2 = 0$. The trace zero subgroup $\mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$ is defined over $\mathbb{F}_{q^6}$, and is generated by $(35u^4, 42u^3)$ (see Figure 2.33). We define the degree $d = 6$ *sextic twist* E' of E as $E' : y^2 = x^3 + 72u^6$, where the

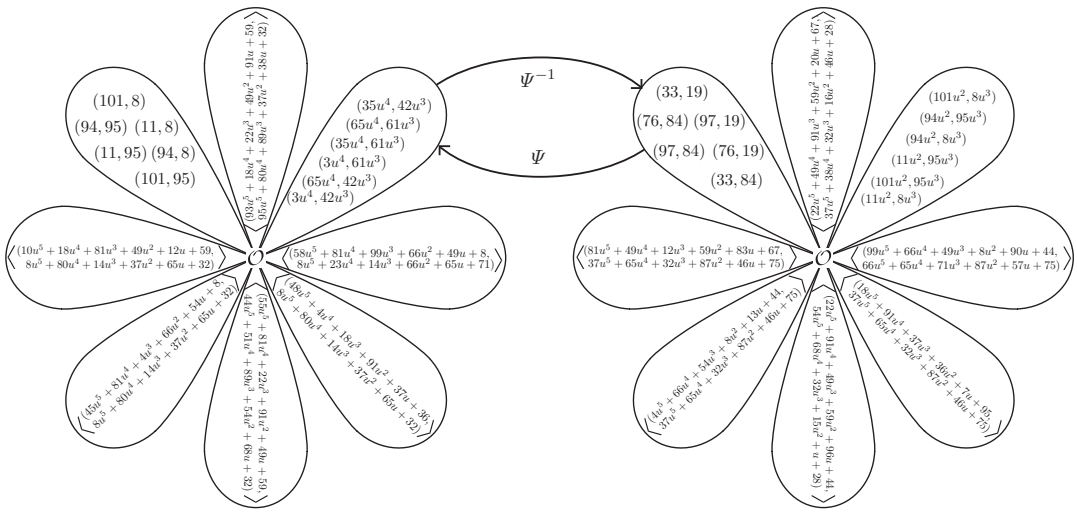


Figure 2.33: E (left) and the (correct) sextic twist E' (right)

back-and-forth isomorphisms are defined as $\Psi : E' \to E$, $(x', y') \mapsto (x'/u^2, y'/u^3)$ and $\Psi : E \to E'$, $(x, y) \mapsto (u^2 x, u^3 y)$. Observe that Ψ^{-1} maps elements in $\mathcal{G}_2 \in E(\mathbb{F}_{q^k})[r] = E(\mathbb{F}_{q^6})[r]$ to elements in $E'(\mathbb{F}_{q^{k/d}})[r] = E'(\mathbb{F}_q)[r]$. Thus, when performing group operations in $\mathbb{G}_2 = \mathcal{G}_2$, we gain the advantage of working over

$\mathbb{F}_q$ instead of $\mathbb{F}_{q^6}$, a dramatic improvement in computational complexity.

In both Example 2.3.8 and Example 2.3.9 above, we had $k = d$, so the twist allowed us to work in the base field $\mathbb{F}_q$, rather than $\mathbb{F}_{q^k}$. In the general case though, the twist will pull computations back into the subfield $\mathbb{F}_{q^{k/d}}$ of $\mathbb{F}_{q^k}$. For example, if the embedding degree was $k = 12$, a quadratic twist ($d = 2$) would allow computations in $\mathbb{G}_2$ to be performed in $\mathbb{F}_{q^6}$ rather than $\mathbb{F}_{q^{12}}$, whilst a sextic twist ($d = 6$) would allow us to instead work in $\mathbb{F}_{q^2}$. Thus, we would clearly prefer the degree d of the twist to be as high as possible. As it turns out, $d = 6$ is the highest degree available on elliptic curves, where the only possibilities are $d \in \{2, 3, 4, 6\}$ [Sil09, Prop. X.5.4]. For $d > 2$, we also require special subclasses of curves that depend on d, so following [Sil09, Prop. X.5.4] (see also [HSV06, Prop. 6, Prop. 8]) we describe all four cases individually. In the general case according to our context, a twist of $E : y^2 = x^3 + ax + b$ is given by $E' : y^2 = x^3 + a\omega^4 x + b\omega^6$, with $\Psi : E' \to E : (x', y') \mapsto (x'/\omega^2, y'/\omega^3)$, $\omega \in \mathbb{F}_{q^k}$. We can only achieve specific degrees d through combinations of zero and non-zero values for a and b.

- $d = 2$ *quadratic twists.* Quadratic twists are available on any elliptic curve, so if $E/\mathbb{F}_q : y^2 = x^3 + ax + b$, then a quadratic twist is given by $E'/\mathbb{F}_{q^{k/2}} : y^2 = x^3 + a\omega^4 x + b\omega^6$, with $\omega \in \mathbb{F}_{q^k}$ but $\omega^2 \in \mathbb{F}_{q^{k/2}}$. Since $\omega^3 \in \mathbb{F}_{q^k}$, the isomorphism $\Psi : E' \to E$ defined by $\Psi : (x', y') \mapsto (x'/\omega^2, y'/\omega^3)$ will take elements in $E'(\mathbb{F}_{q^{k/2}})$ to elements in $E(\mathbb{F}_{q^k})$, whilst Ψ^{-1} will do the opposite.

- $d = 3$ *cubic twists.* Degree $d = 3$ twists can only occur when $a = 0$, so if $E/\mathbb{F}_q : y^2 = x^3 + b$, then $E'/\mathbb{F}_{q^{k/3}} : y^2 = x^3 + b\omega^6$, with $\omega^3, \omega^6 \in \mathbb{F}_{q^{k/3}}$, but $\omega^2 \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^{k/3}}$. Thus, the isomorphism $\Psi : E' \to E$ (defined as usual) will take elements in $E'(\mathbb{F}_{q^{k/3}})$ to elements in $E(\mathbb{F}_{q^k})$, whilst Ψ^{-1} does the opposite.

- $d = 4$ *quartic twists.* Degree $d = 4$ twists are available when $b = 0$, so if $E/\mathbb{F}_q : y^2 = x^3 + ax$, then $E'/\mathbb{F}_{q^{k/4}} : y^2 = x^3 + a\omega^4 x$, with $\omega^4 \in \mathbb{F}_{q^{k/4}}$, $\omega^2 \in \mathbb{F}_{q^{k/2}}$ and $\omega^3 \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^{k/2}}$. Thus, Ψ will move elements in $E'(\mathbb{F}_{q^{k/4}})$ up to elements in $E(\mathbb{F}_{q^k})$, whilst Ψ^{-1} will move elements from $E(\mathbb{F}_{q^k})$ down to $E'(\mathbb{F}_{q^{k/4}})$.

- $d = 6$ *sextic twists.* Sextic twists are only available when $a = 0$, so if $E/\mathbb{F}_q : y^2 = x^3 + b$, then $E'/\mathbb{F}_{q^{k/6}} : y^2 = x^3 + b\omega^6$, with $\omega^6 \in \mathbb{F}_{q^{k/6}}$, $\omega^3 \in \mathbb{F}_{q^{k/3}}$

and $\omega^2 \in \mathbb{F}_{q^{k/2}}$. Thus, Ψ pushes elements in $E'(\mathbb{F}_{q^{k/6}})$ up to $E(\mathbb{F}_{q^k})$, whilst Ψ^{-1} pulls elements from $E(\mathbb{F}_{q^k})$ all the way down to $E'(\mathbb{F}_{q^{k/6}})$.

We make the remark that, for our purposes, a specific twist can only be applied if the curve is of the corresponding form above *and* the embedding degree k has d as a factor. Thus, attractive embedding degrees are those which have any of $d = \{2, 3, 4, 6\}$ as factors, but preferably $d = 4$ or $d = 6$ for increased performance. This will be discussed in detail in Section 2.5. Very fortunately, we will also see in that section that almost all of the popular techniques for constructing curves suitable for pairing computation give rise to curves of the form $y^2 = x^3 + b$ or $y^2 = x^3 + ax$, which facilitate the high-degree twists above.

2.3.4 Section summary

We started by discussing that cryptographic pairings are bilinear maps from two elliptic curve groups to a third (finite field) group $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. We then claimed that, in general, to define a useful pairing on $\mathbb{G}_1$ and $\mathbb{G}_2$, we must be able to define more than one subgroup in the r-torsion of E, where the most cryptographically useful case is that r is a large prime. We then defined the embedding degree k of E (with respect to r), and showed that we must extend the field $\mathbb{F}_q$ to $\mathbb{F}_{q^k}$ in order to find more than one such subgroup. In fact, we showed that $E(\mathbb{F}_{q^k})$ actually contains the entire r-torsion, which has cardinality r^2 and consists of $r+1$ cyclic subgroups of order r. These $r+1$ subgroups (and the existence of maps between them) facilitate several choices for the definitions of $\mathbb{G}_1$ and $\mathbb{G}_2$, which gives rise to four pairing types. We argued that the most popular pairing type is a Type 3 pairing, which sets $\mathbb{G}_1$ and $\mathbb{G}_2$ as the two eigenspaces of the Frobenius endomorphism, namely $\mathbb{G}_1 = \mathcal{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ is the base field subgroup, and $\mathbb{G}_2 = \mathcal{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$ is the trace zero subgroup.

The definitions of the Weil and Tate pairings in the next section inherently justify the claim we made in this section that, in general, the arguments P and Q in the pairing $e(P, Q)$ must come from distinct torsion subgroups.

2.4 Miller's algorithm for the Weil and Tate pairings

This section defines the Weil and Tate pairings and presents Miller's algorithm for computing them. As usual, we state the definitions in our context (on elliptic curves over finite fields), but the more general definitions are analogous (see [Sil09, Gal05]).

Notation. In this section we will use the notation $w_r(P,Q)$ for the (order r) Weil pairing of P and Q and $t_r(P,Q)$ for their (order r) Tate pairing, as this will help when discussing differences and relationships between them. After this chapter though, it will always be clear which pairing we mean and what the value of r is (the largest prime factor of $\#E(\mathbb{F}_q)$), so we will return to the notation most commonly seen in the literature and simply write $e(P,Q)$.

Both pairings make use of a special case of the following fact we recall from Section 2.2: a divisor $D = \sum_P n_P(P)$ is principal (i.e. the divisor of a function) if and only if $\sum_P n_P = 0$ and $\sum_P [n_P]P = \mathcal{O}$ on E. For any $m \in \mathbb{Z}$ and $P \in E$, it follows that there exists a function $f_{m,P}$ with divisor

$$(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O}), \tag{2.17}$$

where we note that for $m = 0$, one can take $f_{0,P} = 1$ with $(f_{0,P})$ the zero divisor. Thus, if $P \in E[r]$, then $f_{r,P}$ has divisor

$$(f_{r,P}) = r(P) - r(\mathcal{O}). \tag{2.18}$$

Observe that $(f_{m+1,P}) - (f_{m,P}) = (P) + ([m]P) - ([m+1]P) - (\mathcal{O})$, which is exactly the divisor of the function $\ell_{[m]P,P}/v_{[m+1]P}$, where $\ell_{[m]P,P}$ and $v_{[m+1]P}$ are the sloped and vertical lines used in the chord-and-tangent addition of the point $[m]P$ and P (see Figure 2.34). This means we can build $f_{m+1,P}$ from $f_{m,P}$ via $f_{m+1,P} = f_{m,P} \cdot \frac{\ell_{[m]P,P}}{v_{[m+1]P}}$.

Example 2.4.1 (Magma script). Let $q = 23$, and consider $E/\mathbb{F}_q : y^2 = x^3 + 17x + 6$ which has $\#E(\mathbb{F}_q) = 30$, and which has $P = (10, 7)$ as a point of order 5. Thus, we are guaranteed the existence of a function $f_{5,P}$ on E with divisor $(f_{5,P}) = 5(P) - 5(\mathcal{O})$. Starting with $m = 2$, we will build $f_{5,P}$ by using $f_{m+1,P} = f_{m,P} \cdot \frac{\ell_{[m]P,P}}{v_{[m+1]P}}$ (note that $(f_{1,P})$ is the zero divisor). The function $f_{2,P}$

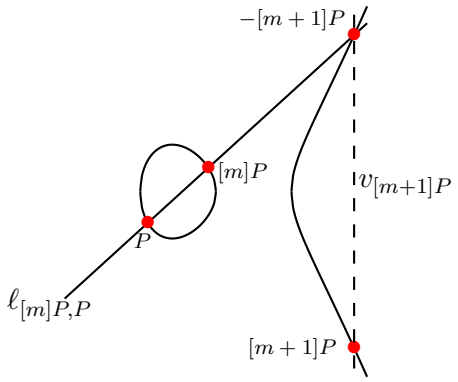


Figure 2.34: $\left(\frac{\ell_{[m]P,P}}{v_{[m+1]P}}\right) = (\ell_{[m]P,P}) - (v_{[m+1]P}) = (P) + ([m]P) - ([m+1]P) - (\mathcal{O})$.

with divisor $(f_{2,P}) = 2(P) - ([2]P) - (\mathcal{O})$ is the tangent line $l_{P,P}$ at P divided by the vertical line $v_{[2]P}$ through $[2]P$, which is $f_{2,P} = \frac{y+2x+19}{x+16}$. We compute the function $f_{3,P}$ as $f_{3,P} = f_{2,P} \cdot \frac{l_{P,[2]P}}{v_{[3]P}}$, where $l_{P,[2]P}$ is the chord through P and $[2]P$ and $v_{[3]P}$ is the vertical line at $[3]P$. Thus, $f_{3,P} = \frac{y+2x+19}{x+16} \cdot \frac{y+x+6}{x+16} = \frac{3y+x^2+9x+19}{x+16}$. Similarly, multiplication by the chord $l_{P,[3]P}$ through P and $[3]P$ and division by the vertical line $v_{[4]P}$ through $[4]P$ will advance us from $f_{3,P}$ to $f_{4,P}$, as $f_{4,P} = f_{3,P} \cdot \frac{l_{P,[3]P}}{v_{[4]P}} = \frac{3y+x^2+9x+19}{x+16} \cdot \frac{y+2x+19}{x+13} = \frac{(x+22)y+5x^2+3x+5}{x+13}$; this function has divisor $(f_{4,P}) = 4(P) - (4P) - 3(\mathcal{O})$. The last update we require is the function with divisor $(P) + (4P) - (5P) - (\mathcal{O})$, which would ordinarily be the quotient of lines in the addition of P and $4P$, but since P has order 5, we know that $P = -4P$, so this function actually has divisor $(P) + (-P) - 2(\mathcal{O})$. Thus, our last update to the function is simply the vertical line at P, i.e. $(x - 10)$, which gives the final function as $f_{5,P} = (x - 10) \cdot \frac{(x+22)y+5x^2+3x+5}{x+13} = (x + 22)y + 5x^2 + 3x + 5$; this function has a zero of order 5 on E at P, and a pole of order 5 on E at $\mathcal{O}$.

2.4.1 The Weil pairing

For a point $P \in E[r]$, the function $f_{r,P}$ with divisor $r(P) - r(\mathcal{O})$ is at the heart of both the Weil and Tate pairing definitions.

Definition 2.2 (The Weil pairing (over finite fields))**.** *Let $P, Q \in E(\mathbb{F}_{q^k})[r]$ and let D_P and D_Q be degree zero divisors with disjoint supports such that $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$. There exist functions f and g such that $(f) = rD_P$ and $(g) = rD_Q$. The Weil pairing w_r is a map*

$$w_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})[r] \to \mu_r,$$

defined as

$$w_r(P,Q) = \frac{f(D_Q)}{g(D_P)}.$$

Among other properties, the Weil pairing is bilinear and non-degenerate. We refer the reader to [Sil09, Ch. III, Prop. 8.1-8.2] for the proofs and full list of properties.

An important point to note is that we can not simply define the Weil pairing as $w_r(P,Q) = f_{r,P}(D_Q)/f_{r,Q}(D_P)$, because $(f_{r,P}) = r(P) - r(\mathcal{O})$ and $(f_{r,Q}) = r(Q) - r(\mathcal{O})$; this corresponds to the divisors $D_P = (P) - (\mathcal{O})$ and $D_Q = (Q) - (\mathcal{O})$, which does not adhere to the requirement that D_P and D_Q have disjoint supports.

Example 2.4.2 (Magma script). Let $q = 23$, and consider $E/\mathbb{F}_q : y^2 = x^3 - x$, which (is supersingular and therefore) has $\#E(\mathbb{F}_q) = q + 1 = 24$. The point $P = (2, 11)$ is a point of order $r = 3$ and the embedding degree with respect to r is $k = 2$. Take $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$ with $i^2 + 1 = 0$, from which we obtain a point Q of order 3 (that is not in $\langle P \rangle$) as $Q = (21, 12i)$, which is actually in the trace zero subgroup, i.e. $\pi(Q) = [q]Q$. Suppose we wish to compute the Weil pairing $w_r(P,Q)$ of P and Q. For illustrative purposes, we will start by computing $f_{r,P}$ and $f_{r,Q}$ and then updating according to the above paragraph. Following the same technique as the last example, we get $f_{r,P}$ and $f_{r,Q}$ as $f_{r,P} = y + 11x + 13$ and $f_{r,Q} = y + 11ix + 10i$, which have divisors $(f_{r,P}) = 3(P) - 3(\mathcal{O})$ and $(g_{r,P}) = 3(Q) - 3(\mathcal{O})$ respectively. We need to find divisors D_P and D_Q that have distinct supports but which are respectively equivalent to $(P) - (\mathcal{O})$ and $(Q) - (\mathcal{O})$. Note that only one of these divisors needs to be updated (so that its support does not contain $\mathcal{O}$), but we will update both in the name of symmetry. Thus, take two more (random) points in $E(\mathbb{F}_{q^2})$ as $R = (17i, 2i + 21)$ and $S = (10i + 18, 13i + 13)$, and set $D_P = (P + R) - (R)$ and $D_Q = (Q + S) - (S)$. We find f as a function with divisor D_P and g as a function with divisor D_Q as $f = f_{r,P}/(\ell_{P,R}/v_{P+R})^3$ and $g = g_{r,Q}/(\ell_{Q,S}/v_{Q+S})^3$ respectively, where $\ell_{P,R}/v_{P+R}$ is the quotient of the chord between P and R and the vertical line through $P + R$ (and similarly for $\ell_{Q,S}/v_{Q+S}$). We can now compute the Weil pairing according to Definition 2.2

as

$$\begin{aligned} w_r(P,Q) &= f(D_Q)/g(D_P) \\ &= \frac{f(Q+S)\cdot g(R)}{f(S)\cdot g(P+R)}. \\ &= 15i+11. \end{aligned}$$

Observe that $(15i+11)^3 = 1$ so $w_r(P,Q) \in \mu_r$. Repeating the whole process with $[2]P$ instead gives $w_r([2]P,Q) = 8i+11 = w_r(P,Q)^2$, or with $[2]Q$ gives $w_r(P,[2]Q) = 8i+11 = w_r(P,Q)^2$, or with both $[2]P$ and $[2]Q$ gives $w_r([2]P,[2]Q) = 15i+11 = w_r(P,Q)^4 = w_r(P,Q)$, which is about as much of the bilinearity of w_r that we can illustrate in this toy example.

2.4.2 The Tate pairing

The formal definition of the Tate pairing requires that only one argument comes from the r-torsion. For our purposes, the other argument can be any point of $E(\mathbb{F}_{q^k})$, but we will soon see that in general it is still advantageous to choose both points from (distinct subgroups in) the r-torsion. In order to define the Tate pairing correctly though, we need to properly define the groups involved. We assume the standard setting that is of most interest to us: $k > 1$, $r \| \#E(\mathbb{F}_q)$ and, since there are r^2 points in the subgroup $E(\mathbb{F}_{q^k})[r]$, we usually have $r^2 \| \#E(\mathbb{F}_{q^k})$. Thus, let $h = \#E(\mathbb{F}_{q^k})/r^2$ be the cofactor that sends points in $E(\mathbb{F}_{q^k})$ to points in $E(\mathbb{F}_{q^k})[r]$. Let $rE(\mathbb{F}_{q^k})$ be the *coset* of points in $E(\mathbb{F}_{q^k})$ defined by

$$rE(\mathbb{F}_{q^k}) = \{[r]P : P \in E(\mathbb{F}_{q^k})\}.$$

The number of elements in $rE(\mathbb{F}_{q^k})$ is h and it contains $\mathcal{O}$; from here we will simply denote this coset as rE. Following [Sco04], we can obtain another distinct coset of $E(\mathbb{F}_{q^k})$ by adding a random element R (not in $E[r]$) to each element of rE. In this way we can obtain precisely r^2 distinct, order h cosets. The quotient group E/rE is the group whose elements are these cosets. We note that elements belonging to each coset do not have the same order, nor do they form a (sub)group. In the quotient group E/rE, points belonging to the same coset (group element) can be used to *represent* the coset. Any two points in the same coset differ from one another by an element in rE, so one can think of

E/rE as the set of equivalence classes of points in $E(\mathbb{F}_{q^k})$ under the equivalence relation $P_1 \equiv P_2$ if and only if $P_1 - P_2 \in rE$ [Gal05, IX.3].

Example 2.4.3 (Magma script). Let $q = 5$, and consider $E/\mathbb{F}_q : y^2 = x^3 - 3$, which has $\#E(\mathbb{F}_q) = 6$. Thus, taking $r = 3$ gives $k = 2$, so take $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$, where $i^2 + 2 = 0$. Further, note that $\#E(\mathbb{F}_{q^2}) = 36 = hr^2$, so $h = 4$, and thus taking $rE = \{[3]P : P \in E(\mathbb{F}_{q^2})\}$ gives $rE = \{\mathcal{O}, (3i + 4, 0), (2i + 4, 0), (2, 0)\}$, with $\#rE = h$. Each of the other 8 cosets in E/rE are shown in Figure 2.35, where we importantly note that each coset has a unique *representative element* that lies in the r-torsion (see Figure 2.36). Consider the coset containing $P_1 = (2i, 4i + 3)$

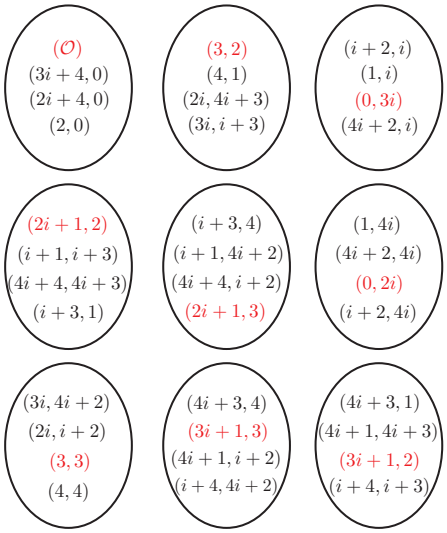


Figure 2.35: The r^2 cosets in the quotient group $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$.

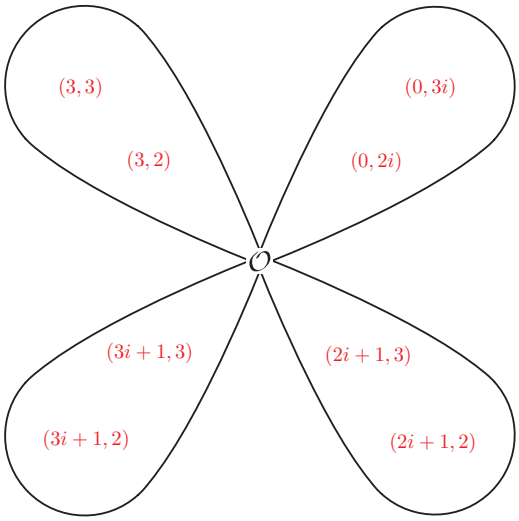


Figure 2.36: The r-torsion, where each $P \in E[r]$ is in a distinct coset of E/rE.

$P_2 = (4, 1)$, $P_3 = (3, 2)$ and $P_4 = (3i, i + 3)\}$. All of the non-trivial pairwise differences are (defined by) $P_1 - P_2 = P_3 - P_4 = (3i + 4, 0)$, $P_1 - P_3 = P_2 - P_4 = (2i + 4, 0)$ and $P_1 - P_4 = P_2 - P_3 = (2, 0)$, which are all in rE.

For our purposes, $E[r]$ and the quotient group E/rE both have r^2 elements[9], but although it was the case in Example 2.4.3, it is not necessarily the case that the elements of $E[r]$ each represent a unique coset of E/rE (see [Gal05, IX.3] for a counterexample). However, if $r^2 \| \#E(\mathbb{F}_{q^k})$, then $E[r] \cap rE = \mathcal{O}$, which means that adding a unique torsion element to all of the elements in rE will generate a unique coset in E/rE. That is, $r^2 \| \#E(\mathbb{F}_{q^k})$ implies that $E[r]$ does represent

[9] In fact, they always have the same number of elements, but there are cases when the cardinality is not r^2 – see [Gal05, IX.3, IX.7.3]

E/rE (see [Gal05, Th. IX.22] for the proof in the supersingular scenario), and this will always be the case for us. This is particularly convenient when it comes to defining the Tate pairing, since the "second" group in the (order r) Tate pairing is E/rE. As we will see after the definition, $E[r]$ representing E/rE allows us to take both groups from the r-torsion, which matches the somewhat simpler Weil pairing group definitions.

We note that although we refer to the following pairing as the Tate pairing throughout, it is often aptly called the Tate-Lichtenbaum pairing [Sil09, XI.9]. This is because Lichtenbaum [Lic69] specialised Tate's more general pairing to the case of Jacobians of curves (over local fields) which facilitates explicit computation [Gal05, IX.3].

Definition 2.3 (The Tate pairing (over finite fields))**.** *Let $P \in E(\mathbb{F}_{q^k})[r]$, from which it follows that there is a function f whose divisor is $(f) = r(P) - r(\mathcal{O})$. Let $Q \in E(\mathbb{F}_{q^k})$ be any representative in any equivalence class in $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$, and let D_Q be a degree zero divisor defined over $\mathbb{F}_{q^k}$ that is equivalent to $(Q)-(\mathcal{O})$, but whose support is disjoint to that of (f). The Tate pairing t_r is a map*

$$t_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \to \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r,$$

defined as

$$t_r(P, Q) = f(D_Q).$$

Again, we remark that among other properties, the Tate pairing is bilinear and non-degenerate. We refer the reader to [Sil09, XI.9] and [Gal05, IX.4] for the proofs and full list of properties.

The quotient group $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r$ is defined as we would expect. Namely, $(\mathbb{F}^*_{q^k})^r$ is a subgroup of $\mathbb{F}^*_{q^k}$ defined as $(\mathbb{F}^*_{q^k})^r = \{u^r : u \in \mathbb{F}^*_{q^k}\}$, so $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^r$ is the set of equivalence classes of $\mathbb{F}^*_{q^k}$ under the equivalence relation $a_1 \equiv a_2$ if and only if $a_1/a_2 \in (\mathbb{F}^*_{q^k})^r$.

Example 2.4.4 (Magma script). We continue with the parameters from Example 2.4.3. Let $P = (3, 2) \in E[r]$ (see Figure 2.35) and let $Q = (i+1, 4i+2) \in E(\mathbb{F}_{q^k})$. The function $f : y + 2x + 2 = 0$ on E has divisor $3(P) - 3(\mathcal{O})$, so to compute the Tate pairing we need to find an appropriate $D_Q \sim (Q) - (\mathcal{O})$ but with $P, \mathcal{O} \notin \text{supp}(D_Q)$. Take R (randomly) as $R = (2i, i + 2)$, and let $D_Q = (Q + R) - (R)$,

where $Q + R = (3i + 1, 2)$. The Tate pairing is computed as

$$t_r(P, Q) = f(D_Q) = \frac{f(Q + R)}{f(R)} = \frac{2 + 2 \cdot (3i + 1) + 2}{(i + 2) + 2 \cdot 2i + 2} = 4i + 4.$$

To illustrate bilinearity, computing $t_r(P, [2]Q)$ with $D_{[2]Q} = ([2]Q + R) - (R)$ where $[2]Q + R = (i + 2, i)$ gives

$$t_r(P, [2]Q) = f(D_{[2]Q}) = \frac{f([2]Q + R)}{f(R)} = \frac{i + 2 \cdot (i + 2) + 2}{(i + 2) + 2 \cdot 2i + 2} = 2i + 4,$$

or computing $t_r([2]P, Q)$, where $\tilde{f} = y + 3x + 3$ has divisor $\tilde{f} = r([2]P) - r(\mathcal{O})$, gives

$$t_r([2]P, Q) = \tilde{f}(D_Q) = \frac{\tilde{f}(Q + R)}{\tilde{f}(R)} = \frac{2 + 3 \cdot (3i + 1) + 3}{(i + 2) + 3 \cdot 2i + 3} = 3i + 2.$$

Note that $t_r(P, Q) = 4i + 4$, $t_r(P, [2]Q) = 2i + 4 = t_r(P, Q)^2$, but $t_r([2]P, Q) = 3i + 2$, i.e. $t_r(P, [2]Q), t_r([2]P, Q) \notin (\mathbb{F}_{q^k}^*)^r$, but $t_r(P, [2]Q)/t_r([2]P, Q) \in (\mathbb{F}_{q^k}^*)^r$, so $t_r(P, [2]Q) \equiv t_r([2]P, Q) \equiv t_r(P, Q)^2$ in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$.

The above example illustrates an important point: in the context of cryptography, the standard Tate pairing has an undesirable property that its output lies in an equivalence class, rather than being a unique value. A necessary attribute for the Tate pairing to be useful in cryptography is that different parties must compute the exact same value under the bilinearity property, rather than values which are the same under the above notion of equivalence. Thus, to be suitable in practice, we must update the definition of the Tate pairing to make sure the mapping produces unique values.

Definition 2.4 (The reduced Tate pairing)**.** *Let P, Q, f and D_Q be as in Definition 2.3. Over finite fields, the reduced Tate pairing T_r is a map*

$$T_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k}) \to \mu_r,$$

defined as

$$\begin{aligned} T_r(P, Q) &= t_r(P, Q)^{\#\mathbb{F}_{q^k}/r} \\ &= f_{r,P}(D_Q)^{(q^k-1)/r}. \end{aligned}$$

Exponentiating elements in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$ to the power of $(q^k-1)/r$ kills r-th powers and sends the paired value to an exact r-th root of unity in μ_r.

From now on we will also take the second argument of the (reduced) Tate pairing from the r-torsion. In fact, we will further assume a Type 3 pairing. Therefore, in the pairing of P and Q, we will assume $P \in \mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ and $Q \in \mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$. One should note that these choices are not restrictions, as far as what values the pairing can take: fixing P and letting Q run through $\langle Q \rangle$ (which has order r) will give each value in μ_r, and vice versa. Thus, for any $\tilde{P}, \tilde{Q}$ pair chosen from anywhere in the torsion, there exists a scalar $0 \leq a \leq r-1$ such that $T_r([a]P, Q) = T_r(P, [a]Q) = T_r(\tilde{P}, \tilde{Q})$.

Example 2.4.5 (Magma script). Let $q = 19$, $E/\mathbb{F}_q : y^2 = x^3 + 14x + 3$, giving $\#E(\mathbb{F}_q) = 20$, so take $r = 5$. The embedding degree is $k = 2$, so let $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$ with $i^2 + 1 = 0$. The points $P = (17, 9)$ and $Q = (16, 16i)$ are in the r-torsion subgroups $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ and $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$ respectively. The Tate pairing of P and Q is $t_r(P, Q) = 7i + 3$, whilst the reduced Tate pairing is $T_r(P, Q) = 15i + 2$. Let $\exp : \mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r \to \mu_r$ be the map defined by the exponentiation $\exp : a \mapsto a^{(q^k-1)/r}$, i.e. $\exp : t_r(P, Q) \mapsto T_r(P, Q)$. Observe the difference between the Tate pairing t_r and reduced Tate pairing T_r for the following computations.

$$\begin{array}{cccc}
t_r(P,Q)^4 & t_r([4]P,Q) & t_r(P,[4]Q) & t_r([2]P,[2]Q) \\
= 3i+7 & = 7i+16 & = 12i+3 & = 2i+14 \\
\Big\downarrow \exp & \Big\downarrow \exp & \Big\downarrow \exp & \Big\downarrow \exp \\
T_r(P,Q)^4 & T_r([4]P,Q) & T_r(P,[4]Q) & T_r([2]P,[2]Q) \\
= 4i+2 & = 4i+2 & = 4i+2 & = 4i+2
\end{array}$$

We note that none of the t_r lie in $(\mathbb{F}_{q^k})^5$, but the quotient of any two of them does lie there, so all the t_r pairings on the top level are equivalent in $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$. On the other hand, T_r ensures that each of the above pairings (that should be equivalent) take exactly the same value in $\mu_r \subset \mathbb{F}_{q^k}$.

From now on, when we say Tate pairing, we mean the reduced Tate pairing T_r in Definition 2.4.

2.4.3 Miller's algorithm

We briefly recap the pairing definitions from the previous two subsections. For the r-torsion points P and Q, the Weil and Tate pairings are respectively com-

puted as $\frac{f_{r,P}(D_Q)}{f_{r,Q}(D_P)}$ and $f_{r,P}(D_Q)^{(q^k-1)/r}$, where the divisors D_P and D_Q are chosen such that their supports are disjoint from the supports of $(f_{r,Q})$ and $(f_{r,P})$ respectively. For any points P and Q belonging to distinct subgroups in $E[r]$, we have already seen how to compute $f_{r,P}(D_Q)$ in the previous subsections, but this was only for very small values of r. In practice r will be huge (i.e. at the very least 2^{160}), and since $f_{r,P}$ is a function of degree approximately r, it is not hard to see that computing this function as we did in the previous examples is impossible. In this subsection we describe Miller's algorithm [Mil04], which makes this computation very feasible. More precisely, the naive method of computing $f_{r,P}(D_Q)$ that we have been using has exponential complexity $O(r)$, whilst the algorithm we are about to describe for this computation has polynomial complexity $O(\log r)$. To put it simply, Miller's algorithm makes pairings practical; without this algorithm, secure cryptographic pairings would only be of theoretical value[10].

We start by referring back to the discussion at the beginning of this section. Following Equation (2.17), we saw that the divisor $(f_{m,P}) = m(P)-([m]P)-(m-1)(\mathcal{O})$ could be updated to the divisor $(f_{m+1}) = (m+1)(P)-([m+1]P)-m(\mathcal{O})$ by adding the divisor $(\ell_{[m]P,P}/v_{[m+1]P}) = (P) + ([m]P) - ([m+1]P) - (\mathcal{O})$; this corresponds to the multiplication of functions $f_{m+1} = f_m \cdot \ell_{[m]P,P}/v_{[m+1]P}$. Starting with $f_{2,P} = 2(P)-([2]P)-(\mathcal{O})$ then, we can repeat this process roughly $r-1$ times to obtain the desired function $f_{r,P} = r(P) - ([r]P) - (r-1)(\mathcal{O}) = r(P) - r(\mathcal{O})$. We note that for the last step (i.e. when $m = r-1$) we have $f_{r-1,P} = (r-1)(P) - ([r-1]P) - (r-2)(\mathcal{O})$, so the required divisor is $(P) + ([r-1]P) - 2(\mathcal{O})$ which corresponds to (a multiplication by!) the vertical line $v_{[r-1]P} = v_{-P} = v_P$; note that this is the same vertical line that appears on the denominator of $\ell_{[r-2]P,P}/v_{[r-1]P}$. Thus, the *pairing evaluation function* $f_{r,P}$ is the product

$$f_{r,P} = \ell_{[r-2]P,P} \cdot \prod_{i=1}^{r-3} \frac{\ell_{[i]P,P}}{v_{[i+1]P}}. \tag{2.19}$$

The first four sloped lines $\ell_{[i]P,P}$ and corresponding vertical lines $v_{[i+1]P}$ from the numerator and denominator of the product in (2.19) are shown in Figure 2.37 and Figure 2.38 respectively. We have seen that the product in (2.19) is (in the

[10]This is no longer entirely true. In 2007 Stange derived an alternative method to Miller's algorithm for efficiently computing the Tate pairing [Sta07], but it is currently less efficient

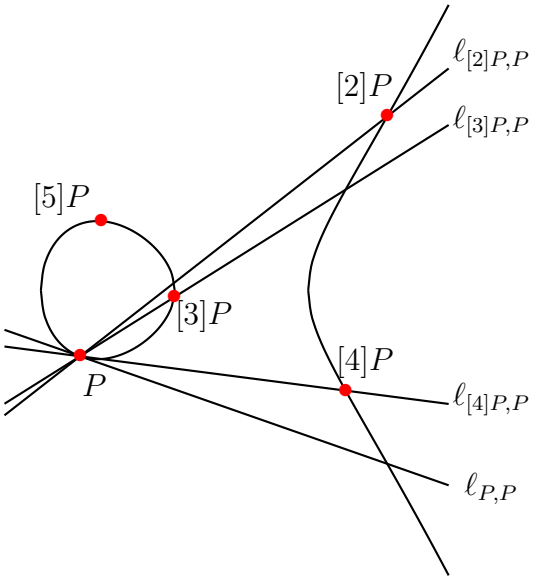


Figure 2.37: The first four sloped lines in the product (2.19).

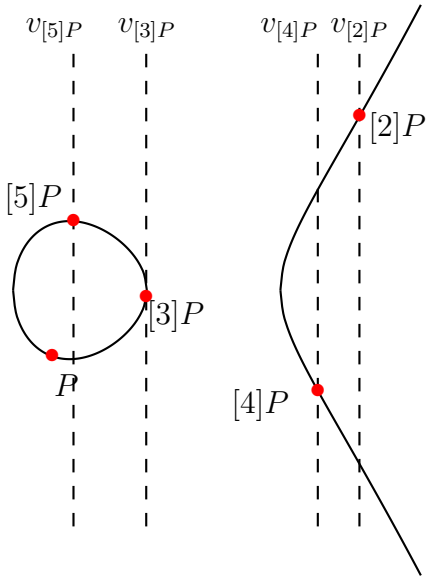


Figure 2.38: The first four vertical lines in the product (2.19).

most naive way) built up incrementally by absorbing each of the $\frac{\ell_{[m]P,P}}{v_{[m+1]P}}$ terms into $f_{m,P}$ to increment to $f_{m+1,P}$, eventually arriving at $f_{r,P}$. Alternatively, it can help to see the divisor sum written out in full, to see the contributions of each of the functions $\frac{\ell_{[i]P,P}}{v_{[i+1]P}}$ in the product all at once.

$$\begin{aligned}
\ell_{P,P}/v_{[2]P}: &\quad (P)+(P)-([2]P)-(\mathcal{O})\\
\ell_{[2]P,P}/v_{[3]P}: &\quad (P)+([2]P)-([3]P)-(\mathcal{O})\\
\ell_{[3]P,P}/v_{[4]P}: &\quad (P)+([3]P)-([4]P)-(\mathcal{O})\\
\vdots &\quad \vdots\quad\vdots\quad\vdots\quad\vdots\\
\ell_{[r-4]P,P}/v_{[r-3]P}: &\quad (P)+([r-4]P)-([r-3]P)-(\mathcal{O})\\
\ell_{[r-3]P,P}/v_{[r-2]P}: &\quad (P)+([r-3]P)-([r-2]P)-(\mathcal{O})\\
\ell_{[r-2]P,P} &\quad (P)+([r-2]P)+(-[r-1]P)-3(\mathcal{O})
\end{aligned}$$

When summing all of the above divisors, most of the inner terms cancel out with one another to leave $(r-1)(P)+(-[r-1]P)-r(\mathcal{O})$, and since $[r-1]P=-P$, we get the divisor of the product being $r(P)-r(\mathcal{O})$.

Roughly speaking, $f_{r,P}=g(x,y)/h(x,y)$, where g and h are degree r functions on E. The above method computes $f_{r,P}$ by successively increasing the

than Miller's algorithm.

degrees of g and h by one each time $f_{m,P}$ is incremented. This is why, when r is (exponentially) large, this naive method has exponential complexity. Miller's algorithm naturally overcomes this through the following observation. The function $f_{m,P}$ has m zeros at P and $(m-1)$ poles at $\mathcal{O}$. Rather than adding one zero and one pole via multiplying $f_{m,P}$ by linear functions, we can double the number of zeros at P and the number of poles at $\mathcal{O}$ if we instead square $f_{m,P}$. Observe that since $(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O})$, then

$$(f_{m,P}^2) = 2m(P) - 2([m]P) - 2(m-1)(\mathcal{O}),$$

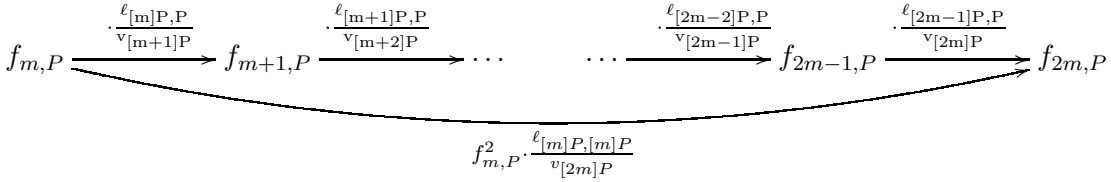
which is almost the same as $f_{2m,P}$, whose divisor is

$$(f_{2m,P}) = 2m(P) - ([2m]P) - (2m-1)(\mathcal{O});$$

the difference between the two divisors being $(f_{2m,P}) - (f_{m,P}^2) = 2([m]P) - ([2m]P) - (\mathcal{O})$, which corresponds to a function with two zeros at $[m]P$, a pole at $[2m]P$ and another pole at $\mathcal{O}$. We have seen such a function many times already; this is simply the quotient of the tangent line at $[m]P$ and the vertical line at $[2m]P$ – the lines used to double the point $[m]P$. Thus, we can advance from $f_{m,P}$ to $f_{2m,P}$ via

$$f_{2m,P} = f_{m,P}^2 \cdot \frac{\ell_{[m]P,[m]P}}{v_{[2m]P}}.$$

We depict the jump from $f_{m,P}$ to $f_{2m,P}$ (as opposed to the naive method of progressing one-by-one) below.



Since, for any m, we can now advance to either $f_{m+1,P}$ or $f_{2m,P}$ quickly, Miller observed that this gives rise to a double-and-add style algorithm to reach $f_{2,r}$ in $O(\log(r))$ steps. However, the degree of $f_{m,P}$ grows linearly in the size of m, so (en route to $m = r$) the function $f_{m,P}$ becomes too large to store explicitly. Thus, the last piece of the puzzle in Miller's derivation of the pairing algorithm was to, at every stage, evaluate $f_{m,P}$ at the given divisor, i.e. $f_{m,P}(D_Q)$. This means that at any intermediate stage of the algorithm we will not be storing

an element of the function field $f_{m,P} \in \mathbb{F}_{q^k}(E)$, but rather its evaluation at D_Q which is the value $f_{m,P}(D_Q) \in \mathbb{F}_{q^k}$. At each stage then, the updates that build the function are evaluated at D_Q before being absorbed into intermediate pairing value that is carried through the routine. This is summarised in Algorithm 2.1 below, where the binary representation of r governs the double-and-add route taken to compute $f_{r,P}(D_Q)$, in an identical fashion to the standard double-and-add routine for scalar multiplications on E (see Example 2.1.10).

Algorithm 2.1 Miller's algorithm.

Input: $P \in E(\mathbb{F}_{q^k})[r]$, $D_Q \sim (Q) - (\mathcal{O})$ with support disjoint from $(f_{r,P})$, and $r = (r_{n-1} \dots r_1 r_0)_2$ with $r_{n-1} = 1$.
Output: $f_{r,P}(D_Q) \leftarrow f$.

1: $R \leftarrow P$, $f \leftarrow 1$.
2: **for** $i = n-2$ down to 0 **do**
3: Compute the line functions $\ell_{R,R}$ and $v_{[2]R}$ for doubling R.
4: $R \leftarrow [2]R$.
5: $f \leftarrow f^2 \cdot \frac{\ell_{R,R}}{v_{[2]R}}(D_Q)$.
6: **if** $r_i = 1$ **then**
7: Compute the line functions $\ell_{R,P}$ and v_{R+P} for adding R and P.
8: $R \leftarrow R + P$.
9: $f \leftarrow f \cdot \frac{\ell_{R,P}}{v_{R+P}}(D_Q)$.
10: **end if**
11: **end for**
12: **return** f.

Miller's algorithm is essentially the straightforward double-and-add algorithm for elliptic curve point multiplication (see Example 2.1.10) combined with evaluations of the functions (the chord and tangent lines) used in the addition process.

Example 2.4.6 (Magma script). We will compute both the Weil and Tate pairings using Miller's algorithm. Let $q = 47$, $E/\mathbb{F}_q : y^2 = x^3 + 21x + 15$, which has $\#E(\mathbb{F}_q) = 51$, so we take $r = 17$. The embedding degree k with respect to r is $k = 4$, thus take $\mathbb{F}_{q^4} = \mathbb{F}_q(u)$ where $u^4 - 4u^2 + 5 = 0$. The point $P = (45, 23)$ has order 17 in $E(\mathbb{F}_q)$ which (because $k > 1$) means $P \in \mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$. The group order over the full extension field is $\#E(\mathbb{F}_{q^4}) = 3^3 \cdot 5^4 \cdot 17^2$, so take $h = 3^3 \cdot 5^4$ as the cofactor. Taking a random point from $E(\mathbb{F}_{q^4})$ and multiplying by h will (almost always) give a point $Q \in E[r]$, but it is likely to land outside of $\mathbb{G}_1 \cup \mathbb{G}_2$, so to move into $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$, we can use the anti-trace map (see Figure 2.25) and take $Q \leftarrow [k]Q - \mathrm{Tr}(Q)$. For example, $Q = (31u^2 + 29, 35u^3 + 11u)$ is one of 17 points in $\mathbb{G}_2$. The Tate pairing is $T_r(P, Q) = f_{r,P}(D_Q)^{(q^k-1)/r}$, whilst

the Weil pairing is $w_r(P,Q) = \frac{f_{r,P}(D_Q)}{f_{r,Q}(D_P)}$. We will illustrate Miller's algorithm to compute $f_{r,P}(D_Q)$, since it appears in both. The binary representation of r is $r = (1,0,0,0,1)_2$. We will take D_Q as $D_Q = ([2]Q) - (Q)$, which clearly has support disjoint to $(f_{r,P})$ and is equivalent to $(Q) - (\mathcal{O})$. The table below shows the stages of Miller's algorithm for computing $f_{r,P}(D_Q)$: it shows the intermediate values of R, and of the function ℓ/v which corresponds to $\frac{\ell_{R,R}}{v_{[2]R}}$ or $\frac{\ell_{R,P}}{v_{R+P}}$ depending on whether we are in the doubling stage (steps 3-5 of Algorithm 2.1) or the addition stage (steps 6-10 of Algorithm 2.1); the table also shows the progression of the paired value f. To complete the Tate pairing, we compute

i/ r_i	steps of Alg. 2.1	point R	update ℓ/v	update at $[2]Q$ = update at Q = $\frac{\ell(D_Q)}{v(D_Q)}$ = $\frac{\ell/v([2]Q)}{\ell/v(Q)}$	paired value f
	1	$(45,23)$			1
3/0	3-5	$(12,16)$	$\frac{y+33x+43}{x+35}$	$\frac{20u^3+21u^2+9u+4}{6u^3+19u^2+36u+33} = 41u^3+32u^2+2u+21$	$41u^3+32u^2+2u+21$
2/0	3-5	$(27,14)$	$\frac{y+2x+7}{x+20}$	$\frac{40u^3+18u^2+38u+9}{39u^3+8u^2+20u+18} = 4u^3+5u^2+28u+17$	$22u^3+27u^2+30u+33$
1/0	3-5	$(18,31)$	$\frac{y+42x+27}{x+29}$	$\frac{29u^3+15u^2+8u+14}{18u^3+32u^2+41u+30} = 6u^3+13u^2+33u+28$	$36u^3+2u^2+21u+37$
0/1	3-5	$(45,24)$	$\frac{y+9x+42}{x+2}$	$\frac{10u^3+3u^2+14u+19}{21u^3+26u^2+25u+20} = 46u^3+45u^2+u+20$	$10u^3+21u^2+40u+25$
	6-10	$\mathcal{O}$	$x+2$	$\frac{7u^2+27}{31u^2+31} = 6u^2+43$	$17u^3+6u^2+10u+22$
	12				$f_{r,P}(D_Q) \leftarrow 17u^3+6u^2+10u+22$

$t_r(P,Q) = f_{r,P}(D_Q)^{(q^k-1)/r} = (17u^3+6u^2+10u+22)^{287040} = 33u^3+43u^2+45u+39$. For the Weil pairing, we require another run of Miller's algorithm, this time reversing the roles of P and Q to compute $f_{r,Q}(D_P) = 2u^2+6u+40$, which gives the Weil pairing as $w_r(P,Q) = \frac{f_{r,P}(D_Q)}{f_{r,Q}(D_P)} = \frac{17u^3+6u^2+10u+22}{2u^2+6u+40} = 22u^3+12u^2+32u+13$. Notice that, in line with Equation 2.19 (and the preceding discussion), the vertical line $x+2=0$ that corresponds to the final addition in this example appears in the denominator of the previous ℓ/v function used for the doubling, and could therefore be cancelled out. We will see that this occurs in general, and is perhaps the least significant of many improvements to Miller's initial algorithm that have accelerated pairings over the last decade. Indeed, in Section 2.6 we will be looking at all the major optimisations to Miller's algorithm, before we proceed into the first chapters of this thesis where we will present our own contributions towards speeding up Algorithm 2.1.

2.4.4 Section summary

We started with the more simple description of the Weil pairing, before moving to the definition of the Tate pairing. This is because both the elliptic curve

groups in the raw definition of the Weil pairing are torsion subgroups, which were discussed at length in the previous section. On the other hand, one of the groups in the general Tate pairing definition required us to introduce the quotient group $E(\mathbb{F}_{q^k})/rE(\mathbb{F}_{q^k})$. However, we soon showed that (for cases of cryptographic interest) it is no problem to represent this quotient group by a torsion subgroup, thereby unifying the group definitions needed for the Weil and Tate pairing and solidifying the choices of $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi - [1])$ and $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [q])$, which will be standard for the remainder of this thesis. We saw that at the heart of both the Weil and Tate pairings is the computation of the pairing evaluation function $f_{r,P}(D)$, where $P \in E$ and D is an appropriately defined divisor on E. We finished the section by presenting Miller's algorithm, which is the first practical algorithm to compute $f_{r,P}(D)$ for cases of cryptographic interest, and which remains the fastest algorithm for computing pairings to date.

2.5 Pairing-friendly curves

To realise pairing-based cryptography in practice, we need two things [Sco07a]:

- efficient algorithms for computing pairings; and
- suitable elliptic curves.

The former was briefly outlined in the last section (and will be taken much further in the next), whilst this section is dedicated to the latter.

2.5.1 A balancing act

Pairings are fundamentally different to traditional number-theoretic primitives, in that they require multiple groups that are defined in different settings. Namely, $\mathbb{G}_1$ and $\mathbb{G}_2$ are elliptic curve groups, whilst $\mathbb{G}_T$ is a multiplicative subgroup of a finite field. All three groups must be secure against the respective instances of the discrete logarithm problem, which means attackers can break the system by solving either the DLP in $\mathbb{G}_T$ or the EDCLP in $\mathbb{G}_1$ or $\mathbb{G}_2$. As we discussed in Section 2.1.1, elliptic curve groups currently obtain much greater security per bit than finite fields; this is because the best attacks on the ECDLP remain generic attacks like Pollard rho [Pol78] which have exponential complexity, whilst the best attacks on the DLP have sub-exponential complexity. In other words, to achieve

the same security, a finite field group needs to have a much greater cardinality than an elliptic curve group. It is standard to state the complexity of asymmetric primitives in terms of the equivalent symmetric key size. For example, the most recent ECRYPT recommendations (see `http://www.keylength.com/en/3/`) say that to achieve security comparable to AES-128 (i.e. 128-bit security), we need an elliptic curve group of approximately 256 bits[11] and a finite field of approximately 3248 bits. We give an example of a curve in the context of pairings for which $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ meet these particular requirements.

Example 2.5.1 (Magma script). Let $E/\mathbb{F}_q : y^2 = x^3 + 14$ be the curve with order $\#E(\mathbb{F}_q)$ having large prime factor r, where q and r are given as

$$\begin{aligned} q = {} & 421943326900167228539204394914103813941511244153259151125138128777531 \\ & 7505016692408034796798044263154903329667 \quad (369 \text{ bits}), \\ r = {} & 223697061107578678950388273657862788561030003896406213385139113737656 \\ & 9980702677867 \quad (271 \text{ bits}). \end{aligned}$$

The embedding degree is $k = 9$, i.e. $q^9 - 1 \equiv 0 \bmod r$. Thus, the two elliptic curve groups $\mathbb{G}_1 \in E[r]$ and $\mathbb{G}_2 \in E[r]$ have an order of 271 bits, which meets the current requirements for 128-bit security. Although $\mathbb{G}_T$ is a subgroup of order r (in $\mathbb{F}^*_{q^k}$), the attack complexity is determined by the full size of the field $\mathbb{F}_{q^9}$, which is 3248 bits, also meeting the requirements for 128-bit security.

We discuss an important point with reference to the above example. Namely, if we were to use primes q and r of the same bit-sizes as Example 2.5.1, but which corresponded to a curve with a larger embedding degree k, then this would not increase the security level offered by the pairing. For example, even though $k = 18$ gives a finite field of 6496 bits, which on its own corresponds to a much harder DLP ($\approx$ 175-bit security), the overall complexity of attacking the protocol remains the same, because the attack complexity of the ECDLP has not changed. Such an increase in k unnecessarily hinders the efficiency of the pairing, since the most costly operations in Miller's algorithm take place in $\mathbb{F}_{q^k}$. Thus, the ideal approach is to optimise the balance between r and $\mathbb{F}_{q^k}$ so that both can be as small as possible whilst simultaneously meeting the particular security level

[11] The "half-the-size" principle between elliptic curve groups and the equivalent asymmetric key size is standard [Sma10, §6.1], since attacks against elliptic curves with order r subgroup have running time $O(\sqrt{r})$. Obtaining the equivalent finite field group size is not as trivial – see [Sma10, §6.2].

required. This was achieved successfully in our example, where $\mathbb{F}_{q^k}$ was exactly the recommended size, and r was only a few bits larger than what is needed to claim 128-bit security.

Nevertheless, we can still obtain a significant improvement on the parameters used in Example 2.5.1; we can keep all three group sizes the same, whilst decreasing the size of the base field $\mathbb{F}_q$. The Hasse bound (see Eq. (2.6)) tells us that the bit-length of $\#E$ and the bit-length of q will be the same. Thus, it is possible that we can find curves defined over smaller fields whose largest prime order subgroup has the same bit-size as that in Example 2.5.1, and whose embedding degree is large enough to offset the decrease in q and therefore that the corresponding full extension field also meets the security requirements. We give a "prime" example.

Example 2.5.2 (Magma script). Let $E/\mathbb{F}_q : y^2 = x^3 + 2$ be the curve with prime order $r = \#E(\mathbb{F}_q)$, where q and r are given as

$$q = 287578801648237372840212049800655234673772199835130985654275192635137696473333173 \quad (271 \text{ bits}).$$

$$r = 287578801648237372840212049800655234673766837197704790989631489840656056071647210 9 \quad (271 \text{ bits}).$$

The embedding degree is $k = 12$, i.e. $q^{12} - 1 \equiv 0 \bmod r$, giving $\mathbb{F}_{q^{12}}$ as a 3248-bit field, which is exactly the same size as the $k = 9$ curve in Example 2.5.1. Thus, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ have orders of the same bit-lengths as before, but using this curve instead means that arithmetic in $\mathbb{F}_q$ will be substantially faster; a 271-bit field in this case, compared to 369-bit field in the last.

In light of the difference between Example 2.5.1 and Example 2.5.2, an important parameter associated with a curve that is suitable for pairings is the ratio between the field size q and the large prime group order r, which we call the ρ-value, computed as

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

Referring back to the two curves above, we have $\rho = \frac{\log q}{\log r} = \frac{369}{271} = 1.36$ in Example 2.5.1, whilst $\rho = \frac{\log q}{\log r} = \frac{271}{271} = 1$ in Example 2.5.2. The ρ-value essentially indicates how much (ECDLP) security a curve offers for its field size, and since we generally prefer the largest prime divisor r of $\#E$ to be as large as possible, $\rho = 1$

is as good as we can get. Indeed, the curve in Example 2.5.2 with $\rho = 1$ belongs to the famous Barreto-Naehrig (BN) family of curves [BN05], which all have $k = 12$ and for which the ratio between the sizes of r and $\mathbb{F}_{q^k}$ make them perfectly suited to the 128-bit security level. This ratio between these group sizes is $\rho \cdot k$ (i.e. $\frac{\log q^k}{\log r} = k \cdot \frac{\log q}{\log r}$), so for commonly used security levels, Figure 2.39 gives the value of $\rho \cdot k$ that balances the current attack complexities of the DLP and ECDLP. Different information security and/or intelligence organisations from around the

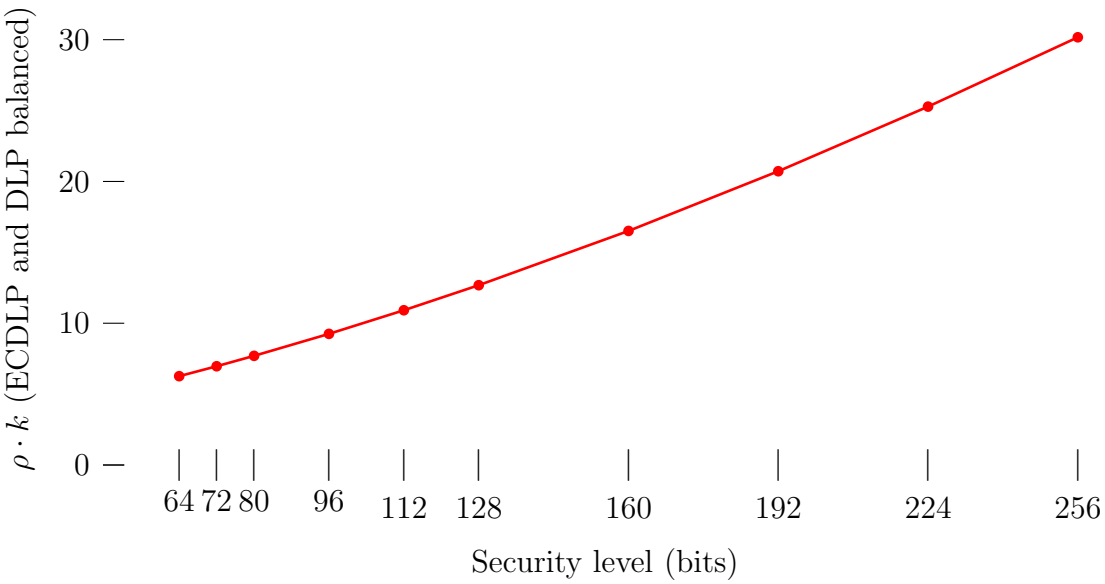


Figure 2.39: The value of $\rho \cdot k$ that balances the complexity of the DLP and ECDLP for commonly used security levels.

globe, such as NIST (the USA) and FNISA (France), give slightly different key size recommendations and complexity evaluations of the algorithms involved; all of this information is conveniently collected at `http://www.keylength.com/`. We have chosen to generate Figure 2.39 according to the numbers in the (most) recent ECRYPT II report [Sma10], which is also summarised there.

Having seen two examples above, we are now in a position to define a *pairing-friendly* curve. Following [FST10], we say that a curve is pairing-friendly if the following two conditions hold:

- there is a prime $r \geq \sqrt{q}$ dividing $\#E(\mathbb{F}_q)$ (i.e. $\rho \leq 2$), and
- the embedding degree k with respect to r is less than $\log_2(r)/8$.

Thus, in their widely cited taxonomy paper, Freeman *et al.* [FST10] consider pairing-friendly curves up to $k = 50$, which is large enough to cover recommended

levels of security for some decades yet.

Balasubramanian and Koblitz [BK98] show that, for q of any suitable cryptographic size, the chances of a randomly chosen curve over $\mathbb{F}_q$ being pairing-friendly is extremely small. Specifically, they essentially show that the embedding degree (with respect to r) of a such a curve is proportional to (and therefore is of the same order as) r, i.e. $k \approx r$. Very roughly speaking, such an argument is somewhat intuitive since (for a random curve) $\#E$ can fall anywhere in the range $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$, so r can be thought of as independent of q, meaning that the order of q in $\mathbb{Z}_r^*$ is random (but see [BK98] for the correct statements). Therefore, imposing that k is small enough to work with elements in $\mathbb{F}_{q^k}$ is an extremely restrictive criterion, so one can not hope to succeed if randomly searching for pairing-friendly curves amongst arbitrary elliptic curves. Thus, in general, pairing-friendly curves require very special constructions.

In Section 2.5.2 we briefly discuss supersingular elliptic curves, which always possess embedding degrees $k \leq 6$ [MOV93, §4], and (so long as $r \geq \sqrt{q}$) are therefore always pairing-friendly. Referring back to Figure 2.39 though, we can see that having $k > 6$ is highly desirable for efficient pairings at the widely accepted security levels, and thus in Section 2.5.3 we focus on the ordinary (non-supersingular) case and outline the constructions that achieve pairing-friendly curves with $k > 6$.

2.5.2 Supersingular curves

Recall from Section 2.3.1 that an elliptic curve E is characterised as supersingular if and only if a distortion map exists on E. There are essentially five types of supersingular curves that are of interest in PBC [Gal05, Table IX.1], but here we will only mention two. This is because we are only concerned with prime fields in this thesis, and the other three are either defined over $\mathbb{F}_{p^2}$, $\mathbb{F}_{2^m}$ or $\mathbb{F}_{3^m}$. As Galbraith mentions, a problem in characteristic 2 and 3 is that there is only a small number of curves and fields to choose from, so there is an element of luck in the search for a curve whose order contains a large prime factor. Another problem in small characteristic is that there exist enhanced algorithms for discrete logarithms (see [Gal05, Ch. IX.13]).

All supersingular curves over large prime fields have $\#E(\mathbb{F}_q) = q+1$, from which it follows that $k = 2$, i.e. regardless of the prime factor $r \neq 2$, $r \mid q+1$ implies $r \nmid q-1$ but $r \mid q^2-1$. We have already seen examples of the two popular

supersingular choices in Section 2.3.1, whose general forms are given in Table 2.1.

q	E	distortion map ϕ	e.g.
2 mod 3	$y^2 = x^3 + b$	$(x, y) \mapsto (\zeta_3 x, y)$, $\zeta_3^3 = 1$	Eg. 2.3.6 (Fig. 2.26)
3 mod 4	$y^2 = x^3 + ax$	$(x, y) \mapsto (-x, iy)$, $i^2 = -1$	Eg. 2.3.7 (Fig. 2.27)

Table 2.1: The two types of popular supersingular curves over prime fields.

We give another example of both cases below, but we choose the parameter sizes to serve another purpose: to show how important it is to employ ordinary curves with higher embedding degrees.

Example 2.5.3 (Magma script). We will choose $q \equiv 11 \bmod 12$ so we can define both examples in Table 2.1 over the same field, but also so that the security of these curves in the context of PBC matches the security of the curve with $k = 12$ in Example 2.5.2. For the ECDLP security to be 128 bits, r still only needs to be 256 bits in size. However, since $k = 2$, for $\mathbb{F}_{q^k}$ to be around 3248 bits, q needs to be around 1624 bits:

$q =$425708693169757088196017853607835113595127103859429924930531263283244403251872949802982860038531930965867890444658222153407204383584492024637762799391807569669124814253270947366226515064812665901907204494611177526596015257984009814596057160388672298355821309046798841446111721495601835913381835880170934319890420895521320439930666405003725309562669243847766834546592867695533445054256132471093279787853214492986394176521193456205570309658462204234557728373615304193316916440130004424612327.

Consider $E_1/\mathbb{F}_q : y^2 = x^3 + 314159$ and $E_2/\mathbb{F}_q : y^2 = x^3 + 265358x$. Both curves have order $\#E_1(\mathbb{F}_q) = \#E_2(\mathbb{F}_q) = q + 1 = hr$, where h is a 1369-bit cofactor and r is the 256-bit prime given as

$r =$5789604461865809771178549250434395392663499233282028201972879200395656482006 3.

The distortion maps are defined over $\mathbb{F}_{q^2} = \mathbb{F}_q(i)$, where $i^2 + 1 = 0$ – see Table 2.1 or Examples 2.3.6 and 2.3.7. The huge size of q stresses the importance of adhering to the optimal ratio of $\rho \cdot k$ suggested by Figure 2.39. A rough

but conservative approximation of the complexity of field multiplications in the 1624-bit field, compared to the 271-bit field in Example 2.5.2 gives a ratio of at least 25 : 1. Referring back to the discussion of pairing types in Section 2.3.2, this gives some idea of the computational price one pays when insisting on the computability of ψ (as well as the other desired properties offered by a Type 1 pairing), rather than adopting a Type 3 pairing and trusting in the heuristics of Chatterjee and Menezes in the absence of such a ψ [CM09, Men09].

We round out this subsection by remarking that although supersingular elliptic curves are limited to $k \leq 6$, Rubin and Silverberg give a practical way to obtain larger values of k using Weil descent [RS02]. Alternatively, one can employ a higher genus supersingular curve to obtain a higher embedding degree [Gal01, RS02]. As Galbraith remarks however, there are severe efficiency limitations in both scenarios, and we achieve faster pairings in practice by using ordinary pairing-friendly elliptic curves [Gal05, Ch. IX.15].

2.5.3 Constructing ordinary pairing-friendly curves

There are three main methods of constructing ordinary pairing-friendly elliptic curves. The two most general methods, the Cocks-Pinch [CP01] and Dupont-Enge-Morain [DEM05] algorithms, produce curves with $\rho = 2$, which is more often than not undesirable when compared to the ρ-values obtained by the third method. Moreover, the third method encompasses all constructions that produce *families* of pairing-friendly elliptic curves, which have been the most successful methods of producing curves that are suitable for current and foreseeable levels of security.

All of the constructions in the literature essentially follow the same idea: fix k and then compute integers t, r, q such that there is an elliptic curve $E/\mathbb{F}_q$ with trace of Frobenius t, a subgroup of prime order r, and an embedding degree k. The *complex multiplication method* (CM method) of Atkin and Morain [AM93] can then be used to find the equation of E, provided the CM discriminant D of E is not too large: D is the square-free part of $4q - t^2$, i.e.

$$Df^2 = 4q - t^2, \tag{2.20}$$

for some integer f. Equation (2.20) is often called the CM equation of E, and by "D not too large" we mean D is less than, say 10^{12} [Sut12].

In 2001, Miyaji, Nakabayashi and Takano [MNT01] gave the first constructions of ordinary pairing-friendly elliptic curves. Their method has since been greatly extended and generalised, but all of the constructions of families essentially followed from their idea, which is aptly named the MNT strategy or MNT criteria [FST10, Gal05]. For some special cases, Miyaji *et al.* used the fact that if k is the (desired) embedding degree, then $r \mid q^k - 1$ implies $r \mid \Phi_k(q)$, since the k-th cyclotomic polynomial $\Phi_k(x)$ is the factor of $x^k - 1$ that does not appear as a factor of any polynomial $(x^i - 1)$ with $i < k$ [Gal05, IX.15.2]. For these cases they were also the first to parameterise *families* of pairing-friendly curves, by writing t, r and q as polynomials $t(x)$, $r(x)$ and $q(x)$ in terms of a parameter x. Miyaji *et al.* focussed on embedding degrees $k = 3, 4, 6$ and assumed that the group order was to be prime, i.e. $r(x) = q(x) + 1 - t(x)$ (from (2.6)). They proved that the only possibilities for $t(x)$ and $q(x)$ (and hence $r(x)$) are

$$\begin{aligned} k = 3: &\quad t(x) = -1 \pm 6x &&\text{and} \quad q(x) = 12x^2 - 1; \\ k = 4: &\quad t(x) = -x, x + 1 &&\text{and} \quad q(x) = x^2 + x + 1; \\ k = 6: &\quad t(x) = 1 \pm 2x &&\text{and} \quad q(x) = 4x^2 + 1. \end{aligned}$$

Example 2.5.4 (Magma script). We kick-start a search for a $k = 4$ (toy) MNT curve with $x = 10$, incrementing by 1 until $q(x) = x^2 + x + 1$ and $r(x) = q(x)+1-t(x)$ (with either of $t(x) = -x$ or $t(x) = x+1$) are simultaneously prime. At $x = 14$, both $q = q(x) = 211$ and $r = r(x) = 197$ (with $t(x) = x+1$) are prime, so we are guaranteed an elliptic curve $E/\mathbb{F}_q$ with r points and embedding degree $k = 4$ (notice $q^4 - 1 \equiv 0 \bmod r$). The CM equation yields $Df^2 = 4q - t^2 = 619$, which itself is prime, so $f = 1$ and thus we seek a curve over $\mathbb{F}_q$ with CM discriminant $D = 619$. The CM method produces one such curve as $E/\mathbb{F}_q : y^2 = x^3 + 112x + 19$. Notice that $\phi_4(q(x)) = q(x)^2 + 1 = (x^2 + 1) \cdot (x^2 + 2x + 2)$, both factors being the possibilities for $r(x)$.

Notice that the toy example above has $\rho = \frac{\log q}{\log r} = \frac{\log 211}{\log 197} = 1.01$. For x of cryptographically large size though, we will get $\rho = 1$ since $q(x) = x^2 + x + 1$ and $r(x) = x^2 + 2x + 2$ or $r(x) = x^2 + 1$ have the same degree. In general parameterised families then, we use the degrees of $q(x)$ and $r(x)$ to state ρ as

$$\rho = \frac{\deg(q(x))}{\deg(r(x))}.$$

A number of works followed the MNT paper and gave useful generalisations of their results. In particular, we mention the work by Barreto *et al.* [BLS02], Scott and Barreto [SB06], and Galbraith *et al.* [GMV07], all three of which obtain more parameterised families by relaxing the condition that the group order is prime and allowing for small cofactors so that $\#E = hr$. Another observation made by Barreto *et al.* that somewhat simplifies the process is the following: $r \mid \Phi_k(q)$ and $q + 1 - t \equiv 0 \bmod r$ combine to give that $\Phi_k(t-1) \equiv 0 \bmod r$ [BLS02, Lemma 1]. Substituting $hr = q + 1 - t$ into the CM equation in (2.20) gives

$$Df^2 = 4hr - (t-2)^2. \tag{2.21}$$

In Section 3.1 of [BLS02], Barreto *et al.* obtain many nice parameterised families for various k by considering a special case of the above equation with $t(x) = x+1$, $D = 3$ and (since $r \mid \Phi_k(x)$) finding $f(x)$ and $m(x)$ to fit

$$3f(x)^2 = 4m(x)\Phi_k(x) - (x-1)^2. \tag{2.22}$$

We note that curves with CM discriminant $D = 3$ are always of the form $y^2 = x^3 + b$. A convenient solution to Equation (2.22) for $k = 2^i \cdot 3$ is $m = (x-1)^2/3$ and $f(x) = (x-1)(2x^4 - 1)/3$, for which we can take $r = \Phi_k(x)$. Taking $i = 3$, we give a cryptographically useful example of a BLS (Barreto-Lynn-Scott) curve with $k = 24$.

Example 2.5.5 (Magma script). Following the above description, the BLS family with $k = 24$ is parameterised as $q(x) = (x-1)^2(x^8-x^4+1)/3+x$, $r(x) = \Phi_{24}(x) = x^8-x^4+1$, $t(x) = x+1$. The family has $\rho = \frac{\deg(q(x))}{\deg(r(x))} = 10/8 = 1.25$ and therefore $\rho \cdot k = 30$. Referring back to Figure 2.39, we see that such a curve gives a nice balance between the sizes of r and q^k (the ECDLP and DLP) for pairings at the 256-bit security level. Indeed, at present this family remains the front-runner for this particular security level [Sco11, CLN11]. To find a curve suitable for this level, we need r to be about 512 bits, and since $\deg(r(x)) = 8$, we will start the search for q, r both prime with a 64-bit value; note that $x \equiv 1 \bmod 3$ makes $q(x)$ an integer, so the first such value is $x = 2^{63} + 2$. After testing a number of incremental $x \leftarrow x + 3$ values, $x = 9223372036854782449$ gives $q(x)$ and $r(x)$ as 629 and 505 bit primes respectively. Since $D = 3$ and $E/\mathbb{F}_q : y^2 = x^3 + b$, i.e. there is only one curve constant, we do not need to use the CM method. Instead, it is usually quicker to try successive values of b until we find the correct curve.

In this case, $b = 1$ gives $E/\mathbb{F}_q : y^2 = x^3 + 1$ as our pairing-friendly $k = 24$ BLS curve.

Barreto *et al.* [BLS02, §3.2] actually give a more general algorithm which, instead of insisting that $t = x + 1$, takes $t = x^i + 1$. Brezing and Weng [BW05] found even more useful families by searching with more general polynomials for $t(x)$. Several constructions followed by looking for parameterisations that satisfy the following conditions which define a family [FST10, Def. 2.7] (also see [Fre06, Def. 2.5]):

(i) $r(x)$ is nonconstant, irreducible, and integer-valued with a positive leading coefficient.

(ii) $r(x) \mid q(x) + 1 - t(x)$.

(iii) $r(x) \mid \Phi_k(t(x) - 1)$.

(iv) The parameterised CM equation $Df^2 = 4q(x) - t(x)^2$ has infinitely many integer solutions (x, f).

Referring to condition (iv) above, we say that a family parameterised by $(t(x),\ r(x),\ q(x))$ is a *complete* family if there exists $f(x) \in \mathbb{Q}[x]$ such that $Df(x)^2 = 4q(x) - t(x)^2$. Otherwise, we say the family is *sparse*. We have already seen a curve belonging to the popular Barreto-Naehrig (BN) family in Example 2.5.2. In the following example we look at the BN parameterisations in terms of the above conditions.

Example 2.5.6 (Magma script). Barreto and Naehrig [BN05] discovered that, for $k = 12$, setting the trace of Frobenius t to be $t(x) = 6x^2 + 1$ gives $\Phi_{12}(t(x) - 1) = \Phi_{12}(6x^2) = (36x^4 + 36x^3 + 18x^2 + 6x + 1)(36x^4 - 36x^3 + 18x^2 - 6x + 1)$. This facilitates the choice of $r(x)$ as the first factor $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$, from which taking $q(x)$ as $q(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ means not only that $r(x) \mid q(x) + 1 - t(x)$ (condition (ii) above), but in fact that $r(x) = q(x) + 1 - t(x)$. Thus, when x is found that makes $r(x)$ and $q(x)$ simultaneously prime, we have a pairing-friendly curve with $k = 12$ that has prime order. Not only is the ρ-value $\rho = 1$ ideal, but there are many more reasons why BN curves have received a great deal of attention [DSD07, PJNB11, AKL+11]. Notice that $D = 3$ and $f(x) = 6x^2 + 4x + 1$ satisfies the CM equation (condition (iv) above), so the BN family is a complete family and BN curves are always of the form $y^2 = x^3 + b$.

The last point of Example 2.5.6 is a crucial one. Referring back to Section 2.3.3, we know that $D = 3$ curves of the form $y^2 = x^3 + b$ admit cubic and sextic twists. Thus, in the case of BN curves where $k = 12$, we can make use of a sextic twist to represent points in $\mathbb{G}_2 \in E(\mathbb{F}_{q^{12}})$ as points in a much smaller subfield on the twist, i.e. in $\Psi^{-1}(\mathbb{G}_2) = \mathbb{G}_2' \in E'(\mathbb{F}_{q^2})$. In general then, when k has the appropriate factor $d \in \{3, 4, 6\}$, we would like to make use of the highest degree twist possible, so we would prefer our pairing-friendly curves to be of the following two forms:

degree d	curve	j-invariant	CM discriminant	field
$3, 6 \mid k$	$y^2 = x^3 + b$	$j(E) = 0$	$D = 3$	$q \equiv 1 \bmod 3$
$4 \mid k$	$y^2 = x^3 + ax$	$j(E) = 1728$	$D = 1$	$q \equiv 1 \bmod 4$

Table 2.2: Pairing-friendly elliptic curves admitting high-degree twists.

See [Sil09][p. 45] for the definition of the j-invariant of an elliptic curve (and the associated calculations); we simply remark that two elliptic curves $E/\mathbb{F}_q$ and $\tilde{E}/\mathbb{F}_q$ are isomorphic over $\overline{\mathbb{F}}_q$ if and only they have the same j-invariant. Due to the preferences in Table 2.2, we will really only be dealing with curves of j-invariants (respectively CM discriminants) $j \in \{0, 1728\}$ (respectively $D \in \{3, 1\}$) in this thesis. In this respect, we are also very fortunate that most of the best constructions of pairing-friendly families have either $D = 1$ or $D = 3$, depending on the embedding degree they target. In general, a severe loss of efficiency is suffered in pairing computations when choosing a curve that does not offer a high-degree twist, so at any particular security level we tend to focus on the curves whose embedding degrees are suitable, both according to Figure 2.39 and which contain $d \in \{3, 4, 6\}$ as a factor [FST10, §8.2]. Besides, as we will see in the next section, there are further efficiency reasons that happily coincide with having $d \mid k$ for $d \in \{3, 4, 6\}$. The equivalence conditions on q in Table 2.2 are to ensure E is ordinary, complementing the supersingular cases in Table 2.1.

Our last example in this section belongs to another complete family from the more recent work of Kachisa, Schaefer and Scott [KSS08], who present record-breaking (in terms of the lowest ρ-value) curves for embedding degrees $k \in \{16, 18, 36, 40\}$.

Example 2.5.7 (Magma script). We choose a KSS curve with $k = 16$, which is parameterised by $t(x) = (2x^5 + 41x + 35)/35$, $q(x) = (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980$ and $r(x) = (x^8 + 48x^4 + 625)/61250$.

This family has $\rho = 5/4$, so referring back to Figure 2.39 we see that $\rho \cdot k = 20$ is a nice fit for pairings at the 192-bit security level. Thus, r should be around 384 bits, so starting our search with x around 2^{50} should do the trick (we add the extra two bits to account for the 16-bit denominator of $r(x)$). The polynomials for $q(x)$ and $t(x)$ can only take on integers if $x \equiv \pm 25 \bmod 70$, so we start with $x \equiv 2^{50} + 21 \equiv 25 \bmod 70$ and iterate accordingly. We soon arrive at $x = 1125899907533845$, which gives a 491-bit q as

$$\begin{aligned} q =& 3340194518359587075607904514504348578130581647867654217642899810042867643534741048241225178436682317003010155280705836842596368221341280505964970897, \end{aligned}$$

and a 385-bit prime factor r of $\#E(\mathbb{F}_q)$ as

$$\begin{aligned} r =& 4215918189011304280250800671237881596873006793850195934448558095361634092780222932018149564359414764607793390912163 3. \end{aligned}$$

Again, we do not need the CM method to find the curve: we simply start with $a = 1$ in $y^2 = x^3 + ax$ and increment until we find $a = 3$ which gives the correct curve as $E/\mathbb{F}_q : y^2 = x^3 + 3x$. E has embedding degree 16 with respect to r, so the full extension field $\mathbb{F}_{q^k}$ is 7842 bits.

We finish this section with two important remarks, the first of which is consequential all the way through this thesis.

Remark 2.5.1 (Curves for ECC vs. curves for PBC). At the highest level, finding curves that are suitable for ECC really imposes only one condition on our search, whilst finding curves that are suitable for PBC imposes two: in ECC we only look for curves with large prime order subgroups, whilst in PBC we have the added stipulation in that we also require a low embedding degree. Whilst one can search for suitable curves for ECC by checking "random" curves until we come across one with almost prime order, in PBC we require very special constructions (like all those discussed in this section) that also adhere to the extra criterion – as we have already discussed, we can not expect to find any pairing-friendly curves by choosing curves at random [BK98]. A major consequence is that in ECC we can specify the underlying field $\mathbb{F}_q$ however we like before randomly looking for a suitable curve over that field. In this case fields can therefore be chosen to take advantage of many low-level optimisations; for

example, Mersenne primes achieve very fast modular multiplications which blows out the relative cost of inversions. On the other hand, in PBC we are confined to the values taken by the polynomials $q(x)$ and have limited control over the prime fields we find. Thus, we are not afforded the luxury of many low-level optimisations and this drastically affects the ratios between field operations (inversions/multiplications/squarings/additions). For example, whilst $\mathbb{F}_q$-inversions in ECC are commonly reported to cost more than 80 $\mathbb{F}_q$-multiplications, the ratio in the context of PBC is nowhere near as drastic [LMN10, AKL+11]. This means we often have to rethink trade-offs between field operations that were originally popularised in ECC.

Remark 2.5.2 (Avoiding pairing-friendly curves in ECC). In the previous remark we said that in ECC we only need to satisfy one requirement (the large prime subgroup), but this is not the full story. In fact, in this context we prefer to choose curves that are strictly not pairing-friendly. After all, in ECC there is no need for a low embedding degree, so choosing a curve that (unnecessarily) has one gives an adversary another potential avenue for attack. Indeed, exploiting curves with low embedding degrees in the context of ECC was the first use of pairings in cryptography – the famous Menezes-Okamato-Vanstone (MOV) [MOV93] and Frey-Rück (FR) [FR94] attacks. Thus, so long as we avoid supersingular curves, the heuristic argument [BK98] tells us that the curves we choose at random will have enormous embedding degrees with overwhelmingly high probability, so this is not a restriction in the sense of Remark 2.5.1.

2.5.4 Section summary

We stressed the importance of finding elliptic curves with large prime order subgroups and small embedding degrees, i.e. pairing-friendly curves. We showed that supersingular curves, whilst easy to find, severely limit the efficiency of pairing computations, particularly at moderate to high levels of security, because they are confined to $k \leq 6$ (and $k \leq 2$ over prime fields). Thus, we turned our focus to the more difficult task of constructing ordinary pairing-friendly elliptic curves, and summarised many landmark results that have enhanced this arena over the last decade. In particular, we gave examples of some of the most notable families of pairing-friendly elliptic curves; such families are the main focus of Chapter 5 and Chapter 6 of this thesis.

2.6 The state-of-the-art

This section summarises the evolution of pairing computation over the last decade. We illustrate the landmark achievements that accelerated early implementations of pairings from "a few minutes" [Men93][12] into current implementations that take less than a millisecond [AKL+11].

Initial improvements in pairing computations were spearheaded by evidence that computing the Tate pairing $f_{r,P}(D_Q)^{(q^k-1)/r}$ is more efficient than computing the Weil pairing $f_{r,P}(D_Q)/f_{r,Q}(D_P)$. At first glance it seems that comparing the two computations amounts to comparing an exponentiation by $(q^k-1)/r$ to a (second) run of Miller's algorithm $f_{r,Q}(D_P)$, and indeed, at levels of security up to 128 bits, this comparison does favour the Tate pairing (cf. [SCA06, Tab. 1-5], [Sco07c]). However, as we will see in Section 2.6.1, exponentiating by $(q^k-1)/r$ actually facilitates many "Tate-specific" optimisations within the associated Miller loop. It is these enhancements that gave the field of pairing computation its first big boost.

2.6.1 Irrelevant factors (a.k.a. denominator elimination)

In this subsection we will work our way to a refined version of Miller's algorithm for pairings over large prime fields, which is mostly due to improvements suggested by Barreto, Kim, Lynn and Scott [BKLS02], and also partly due to Galbraith, Harrison and Soldera [GHS02]. Thus, it is often referred to as the BKLS algorithm [Sco05a, WS07], or sometimes as the BKLS-GHS algorithm [Sco05b, BGOS07]. Our exposition will make use of twisted curves, which we discussed in Section 2.3.3 and the employment of which is originally due to Barreto, Lynn and Scott [BLS03]. The early works that included Barreto, Lynn and Scott are also culminated in [BLS04].

We start with an observation that allows us to conveniently replace the divisor D_Q with the point Q in the Tate pairing definition. Namely, so long as $k > 1$ and P and Q are linearly independent, then $f_{r,P}(D_Q)^{(q^k-1)/r} = f_{r,P}(Q)^{(q^k-1)/r}$ [BKLS02, Th. 1]. This saves the hassle of defining a divisor equivalent to $D_Q = (Q) - (\mathcal{O})$ with support disjoint to $(f_{r,P})$, but more importantly allows us to simply evaluate the intermediate Miller function at the point Q (rather than two

[12] As Scott says however, this comparison is unfair – in 1993 there was no incentive to try and optimise the computation past what was needed to apply the MOV attack [MOV93].

points) in each iteration of Algorithm 2.1.

Example 2.6.1 (Magma script). We reuse the parameters from Example 2.4.6 so a comparison between intermediate values is possible. Thus, let $q = 47$, $E/\mathbb{F}_q : y^2 = x^3 + 21x + 15$, $\#E(\mathbb{F}_q) = 51$, $r = 17$, $k = 4$, $\mathbb{F}_{q^4} = \mathbb{F}_q(u)$ with $u^4 - 4u^2 + 5 = 0$, $P = (45, 23) \in \mathbb{G}_1$ and $Q = (31u^2 + 29, 35u^3 + 11u) \in \mathbb{G}_2$. Thus, the Tate pairing is $e(P, Q) = f_{r,P}(Q)^{(q^k-1)/r} = (32u^3 + 17u^2 + 43u + 12)^{287040} = 33u^3 + 43u^2 + 45u + 39$, which is the same value we got when instead computing $f_{r,P}(D_Q) = f_{r,P}([2]Q)/f_{r,P}(Q)$ in Example 2.4.6. When comparing the fifth columns of both tables, one should keep in mind that the numerator and denominator of the fractions in Example 2.4.6 were themselves both computed as fractions. Indeed, updates in this example are just the denominator of the updates in Example 2.4.6, which gives an indication of how advantageous it is to evaluate the pairing functions at one point (e.g. Q), rather than at a divisor consisting of multiple points (e.g. $([2]Q) - (Q)$). Notice that the values $f_{r,P}(D_Q)$ and $f_{r,P}(Q)$ output after the Miller loops in both examples are not the same, but the *final exponentiation* maps them to the same element in μ_{17}. This is because $f_{r,P}(D_Q)$ and $f_{r,P}(Q)$ lie in the same coset of $(\mathbb{F}_{q^k}^*)^r$ in $\mathbb{F}_{q^k}^*$, i.e. they are the same element in the quotient group $\mathbb{F}_{q^k}^*/(\mathbb{F}_{q^k}^*)^r$.

i/ r_i	steps of Alg. 2.1	point R	update ℓ/v	update at Q $\ell(Q)/v(Q)$	paired value f
	1	$(45, 23)$			1
3/0	3-5	$(12, 16)$	$\frac{y+33x+43}{x+35}$	$\frac{35u^3+36u^2+11u+13}{31u^2+17} = 6u^3 + 19u^2 + 36u + 33$	$6u^3 + 19u^2 + 36u + 33$
2/0	3-5	$(27, 14)$	$\frac{y+2x+7}{x+20}$	$\frac{35u^3+15u^2+11u+18}{31u^2+2} = 39u^3 + 8u^2 + 20u + 18$	$11u^3 + 17u^2 + 24u + 4$
1/0	3-5	$(18, 31)$	$\frac{y+42x+27}{x+29}$	$\frac{35u^3+33u^2+11u+23}{31u^2+11} = 18u^3 + 32u^2 + 41u + 30$	$22u^3 + 34u^2 + 5u + 10$
0/1	3-5	$(45, 24)$	$\frac{y+9x+42}{x+2}$	$\frac{35u^3+44u^2+11u+21}{31u^2+31} = 21u^3 + 26u^2 + 25u + 20$	$8u^3 + 22u^2 + 5u + 27$
	6-10	$\mathcal{O}$	$x + 2$	$= 31u^2 + 31$	$32u^3 + 17u^2 + 43u + 12$
	12			$f_{r,P}(Q) \leftarrow 32u^3 + 17u^2 + 43u + 12$	

We are now in a position to describe the important *denominator elimination* optimisation. Barreto *et al.* were the first to notice that $q - 1 \mid (q^k - 1)/r$ [BKLS02, Lemma 1], since if $r \mid q-1$ then the embedding degree would be $k = 1$. This allows us to write the final exponent as $(q^k - 1)/r = (q - 1) \cdot c$, which gives $f_{r,P}(Q)^{(q^k-1)/r} = (f_{r,P}(Q)^{q-1})^c$, meaning that any elements of $\mathbb{F}_q$ contributing to $f_{r,P}(Q)$ will be mapped to one under the final exponentiation. Thus, one can freely multiply or divide $f_{r,P}(Q)$ by an element of $\mathbb{F}_q$ without affecting the pairing value [BKLS02, Corr. 1]. When working over supersingular curves with $k = 2$, the x-coordinate of Q is defined over $\mathbb{F}_q$ (see any of Examples 2.3.6, 2.3.7 ,2.3.8,

2.4.3). Therefore, the vertical lines appearing on the denominators of Miller's algorithm for the Tate pairing are entirely defined over $\mathbb{F}_q$: the line is a function $x - x_R$ that depends on $P \in E(\mathbb{F}_q)[r]$, which is evaluated at $x_Q \in \mathbb{F}_q$. Thus, in this case the contribution of (each of) the denominators to $f_{r,P}(Q)$ ends up being mapped to 1 under the final exponentiation, so these denominators (the v's in the ℓ/v's – see Steps 5 and 9 in Algorithm 2.1) can be removed from the Miller loop.

For ordinary curves with $k > 2$ however, the x-coordinate of Q will no longer be in the base field $\mathbb{F}_q$, but in some proper subfield $\mathbb{F}_{q^e}$ of $\mathbb{F}_{q^k}$, where $e = k/d$ and d is the degree of the twist employed[13] – see Section 2.3.3. Here it helps to assume that k is even, i.e. $k = 2\ell$, so that (at the very least) we can take $Q = (x_Q, y_Q)$ where $x_Q \in \mathbb{F}_{q^\ell}$ is such that $y_Q \in \mathbb{F}_{q^k} \setminus \mathbb{F}_{q^\ell}$. Thus, when advancing beyond $k = 2$ supersingular curves, Barreto *et al.* generalised the original statement to facilitate the same trick. Namely, that $q^e - 1 \mid (q^k - 1)/r$ for any proper factor $e \mid k$ [BLS03, Lemma 5, Corr. 2], so denominators can be omitted from computations in general.

Example 2.6.2 (Magma script). Again, we will continue on from Example 2.6.1 for the sake of a convenient comparison. We simply give an updated table that details the intermediate Miller functions and pairing values subject to denominator elimination. Therefore, $e(P,Q) = f_{r,P}(Q)^{(q^k-1)/r} = (9u^3 + 10u^2 + 32u + 36)^{(q^k-1)/r} = 33u^3 + 43u^2 + 45u + 39$, which agrees with the Tate pairing value in Examples 2.4.6 and 2.6.1. Notice again that the value output from the Miller loop is not equal to either of the values output in 2.4.6 or 2.6.1, but rather that all three are equivalent under the relation $a = b$ if $a/b \in (\mathbb{F}_{q^k}^*)^r$.

$i/$ r_i	steps of Alg. 2.1	point R	update ℓ	update at Q $\ell(Q)$	paired value f
	1	$(45, 23)$			1
3/0	3-5	$(12, 16)$	$y + 33x + 43$	$35u^3 + 36u^2 + 11u + 13$	$35u^3 + 36u^2 + 11u + 13$
2/0	3-5	$(27, 14)$	$y + 2x + 7$	$35u^3 + 15u^2 + 11u + 18$	$44u^3 + 34u^2 + 3u + 44$
1/0	3-5	$(18, 31)$	$y + 42x + 27$	$35u^3 + 33u^2 + 11u + 23$	$5u^3 + 24u^2 + 21u + 24$
0/1	3-5	$(45, 24)$	$y + 9x + 42$	$35u^3 + 44u^2 + 11u + 21$	$21u^3 + 36u^2 + 9u + 25$
	6-10	$\mathcal{O}$	$x + 2$	$31u^2 + 31$	$9u^3 + 10u^2 + 32u + 36$
	12	$f_{r,P}(Q) \leftarrow 9u^3 + 10u^2 + 32u + 36$			

Notice that the denominators that

[13] When $d = 3$ cubic twists are able to be employed for odd k, it is the y-coordinate of Q that is in the subfield; we will treat this in Chapter 2.3.

were on lines 5 and 9 have now gone (under the assumption that k is even), and that the second input is now the point Q, rather than a divisor equivalent to D_Q. Further notice that we have necessarily include the final exponentiation in Algorithm 2.2 since this is what facilitates the modifications. We have also assumed a Type 3 pairing so the coordinates of P and Q lie in fields that allow for denominator elimination. Recall from the discussion at the end of Example 2.4.6, or from Example 2.6.2, that the vertical line joining $(r-1)P = -P$ and P in the last iteration can also be omitted. Thus, an optimised Tate pairing computation will execute the main loop from $i = n-2$ to $i = 1$ before performing a "doubling-only" iteration to finish; we left the main loop to $i = 0$ for simplicity.

Algorithm 2.2 The BKLS-GHS version of Miller's algorithm for the Tate pairing.

Input: $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ (Type 3 pairing) and $r = (r_{n-1} \dots r_1 r_0)_2$ with $r_{n-1} = 1$.
Output: $f_{r,P}(Q)^{(q^k-1)/r} \leftarrow f$.

1: $R \leftarrow P$, $f \leftarrow 1$.
2: **for** $i = n-2$ down to 0 **do**
3: Compute the sloped line function $\ell_{R,R}$ for doubling R.
4: $R \leftarrow [2]R$.
5: $f \leftarrow f^2 \cdot \ell_{R,R}(Q)$.
6: **if** $r_i = 1$ **then**
7: Compute the sloped line function $\ell_{R,P}$ for adding R and P.
8: $R \leftarrow R + P$.
9: $f \leftarrow f \cdot \ell_{R,P}(Q)$.
10: **end if**
11: **end for**
12: **return** $f \leftarrow f^{(q^k-1)/r}$.

2.6.2 Projective coordinates

Although the optimisations described in the previous section removed the denominators in Step 5 and Step 9 of Algorithm 2.2, $\mathbb{F}_q$-inversions are still apparent in the routine since the affine explicit formulas for the elliptic curve group operations (see Eq. (2.4) and (2.5)) require them. The penalty for performing field inversions in PBC is not as bad as it is in ECC (more on this later), but in any case inversions are still much more costly than field multiplications. In this subsection, we employ the same techniques to avoid field inversions as we did in the context of ECC in Example 2.1.11. Namely, we show how Algorithm 2.2 can become inversion-free if we adopt projective coordinates. In the early days the

situation for projective coordinates in the context of pairings was perhaps a little unclear [Gal05, IX.14], but nowadays all of the record-breaking implementations (at least up to the 128-bit security level) have exploited the savings offered by working in projective space.

The potential of projective coordinates was mentioned in passing in the early landmark papers [BKLS02, §3.2], [GHS02], but the first detailed investigation was by Izu and Tagaki [IT02]. As Galbraith mentions [Gal05, IX.14], the analysis in [IT02] is misleading, however projective coordinates did not wait too long before more accurate expositions that also endorsed their usefulness surfaced [CSB04, Sco05a]. Chapter 3 of this thesis is dedicated to pushing the limits in this direction, so we will save much of the discussion until then, but for now it is useful to see an inversion-free example of Miller's algorithm.

Example 2.6.3 (Magma script). In the context of standard ECC operations, we gave the (homogeneous) projective point addition formulas in Example 2.1.11. Thus, here we will give the homogeneous doubling formulas for computing $(X_{[2]R} : Y_{[2]R} : Z_{[2]R}) = [2](X_R : Y_R : Z_R)$ on $E/\mathbb{F}_q : Y^2Z = X^3 + aXZ^2 + bZ^3$ in Step 4 of Algorithm 2.2, together with the formulas for computing the line function $\ell_{R,R}(Q)$ in Step 3. The affine doubling formulas in Equation (2.5) are moved into homogeneous projective space via the substitution $x = X/Z$ and $y = Y/Z$, which gives:

$$\lambda = \frac{3X_R^2 + Z_R^2}{2Y_RZ_R}; \qquad \nu = -\frac{3X_R^3 + X_RZ_R^2 - 2Y_R^2Z_R}{2Y_RZ_R^2};$$
$$\frac{X_{[2]R}}{Z_{[2]R}} = \frac{-8X_RY_R^2Z_R + 6X_R^2Z_R^2 + 9X_R^4 + Z_R^4}{4Y_R^2Z_R^2};$$
$$\frac{Y_{[2]R}}{Z_{[2]R}} = -\frac{8Y_R^4Z_R^2 + Z_R^6 - 12X_RZ_R^3Y_R^2 - 36X_R^3Z_RY_R^2 + 27Z_R^2X_R^4 + 9Z_R^4X_R^2 + 27X_R^6}{8Y_R^3Z_R^3},$$

where $\ell : y - (\lambda x + \nu)$ is still an affine line tangent to E at the point R. It is again the ability to multiply by factors in proper subfields of $\mathbb{F}_{q^k}$ that allows us to arrive at an inversion-free routine. Namely, we clear the denominators of λ and ν through multiplication by $2Y_RZ_R^2$, so the line ℓ becomes

$$\ell : (2Y_RZ_R^2) \cdot y - ((3X_R^2Z_R + Z_R^3) \cdot x - (3X_R^3 + X_RZ_R^2 - 2Y_R^2Z_R)),$$

which will be evaluted at $y = y_Q$ and $x = x_Q$. Note that since Q remains fixed throughout the routine, there is no need to cast it into projective space. Finally, setting $Z_{[2]R} = 8Y_R^3Z_R^3$ and updating the numerator of $X_{[2]R}$ above allows us to

compute $(X_{[2]R} : Y_{[2]R} : Z_{[2]R})$ from $(X_R : Y_R : Z_R)$ without any $\mathbb{F}_q$-inversions. Thus, we have an inversion-free way to proceed through the Miller doubling stage (Steps 3-5 of Algorithm 2.2), and performing the analogous procedure for the Miller addition stage (Steps 7-9) will give an inversion-free Miller loop. We make no attempt to optimise or delve further into the above computations in this example; this is the purpose of Chapter 3.

2.6.3 Towered extension fields

This subsection discusses efficient methods of constructing the full extension field $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$, where the ultimate goal is to minimise the cost of the arithmetic in $\mathbb{F}_{q^k}$. Indeed, the majority of operations within the pairing algorithm take place in the full extension field, which is far more expensive to work in than its proper subfields, so the complexity of Miller's algorithm heavily depends on the complexity of the associated $\mathbb{F}_{q^k}$-arithmetic.

So far we have been using one irreducible degree k polynomial to construct $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. This has been satisfactory, since our small examples have mostly had embedding degrees $k = 2$ or $k = 3$, where we have no other option but to use polynomials of degree two and three to respectively construct $\mathbb{F}_{q^k}$. However, for large values of k, which will be composite in all cases of interest to us, there is an a natural alternative which turns out to be much faster. This idea was first put forward by Koblitz and Menezes [KM05], who proposed using embedding degrees of the form $k = 2^i 3^j$ and building up to $\mathbb{F}_{q^k}$ using a series of quadratic and cubic extensions that successively *tower* up the intermediate fields. For such k, they show that if $q \equiv 1 \bmod 12$ and if α is neither a square or cube in $\mathbb{F}_q$, then the polynomial $x^k - \alpha$ is irreducible in $\mathbb{F}_q[x]$ [KM05, Th. 1]. This means that the tower can be constructed by a sequence of Kummer extensions: this involves successively adjoining the square root or cube root α, then the square root or cube root of that, and so on.

Example 2.6.4 (Magma script). Let $q = 97$, and consider constructing $\mathbb{F}_{q^{12}}$ using $\alpha = 5$ which is a non-square and non-cube in $\mathbb{F}_q$, so that $\mathbb{F}_{q^{12}}$ can be constructed directly as $\mathbb{F}_{q^{12}} = \mathbb{F}_q[X]/(X^{12} - \alpha)$. Choosing instead a tower of quadratic and cubic extensions, we could construct $\mathbb{F}_{q^{12}}$ as

$$\mathbb{F}_q \xrightarrow{\beta^2-\alpha} \mathbb{F}_{q^2} \xrightarrow{\gamma^3-\beta} \mathbb{F}_{q^6} \xrightarrow{\delta^2-\gamma} \mathbb{F}_{q^{12}}.$$

We show a random element in $\mathbb{F}_{q^{12}}$:

$$\left((79\beta+63)\gamma^2+(29\beta+63)\gamma+(38\beta+27)\right)\delta+(63\beta+22)\gamma^2+(93\beta+10)\gamma+75\beta+10.$$

Observe what happens if, instead of performing multiplications in $\mathbb{F}_{q^{12}}$ over $\mathbb{F}_q$, we start by performing multiplications over $\mathbb{F}_{q^6}$. Writing $a, b \in \mathbb{F}_{q^{12}}$ over $\mathbb{F}_{q^6}$ gives $a = a_0+a_1\delta$ and $b = b_0+b_1\delta$, with $a_0, a_1, b_0, b_1 \in \mathbb{F}_{q^6}$. Thus, $a\cdot b = (a_0b_0-a_1b_1\gamma)+(a_0b_1+a_1b_0)\delta$, where each of the components inside the parentheses are in $\mathbb{F}_{q^6}$. To perform each of the multiplications in $\mathbb{F}_{q^6}$, we then work over $\mathbb{F}_{q^2}$, so for example we would need to compute a multiplication between $a_0 = a_{0,0} + a_{0,1}\gamma + a_{0,2}\gamma^2$ and $b_0 = b_{0,0} + b_{0,1}\gamma + b_{0,2}\gamma^2$, where each component $a_{0,i}$ and $b_{0,i}$ is in $\mathbb{F}_{q^2}$. In this way the operations filter down the tower until we are performing multiplications in $\mathbb{F}_q$.

The computational advantage of adopting a tower of extensions may not be immediately evident. Namely, suppose we were to analyse the complexity of the $\mathbb{F}_{q^{12}}$ multiplication in Example 2.6.4. If we were to employ the naive "schoolbook" method of multiplying two extension field elements, which operates componentwise, then an $\mathbb{F}_{q^{12}}$ multiplication computed directly over $\mathbb{F}_q$ would cost 144 $\mathbb{F}_q$ multiplications. If we instead descend down the tower employing schoolbook multiplication, then an $\mathbb{F}_{q^{12}}$ multiplication would cost 4 $\mathbb{F}_{q^6}$ multiplications, each of which would cost 9 $\mathbb{F}_{q^2}$ multiplications, with each of these costing 4 multiplications in $\mathbb{F}_q$, giving $4 \cdot 9 \cdot 4 = 144$ base field multiplications in this case too. However, one of the reasons that the towered approach betters a direct extension to $\mathbb{F}_{q^k}$ is because there exist much better (than schoolbook) methods of performing arithmetic in quadratic and cubic extensions. Specifically, the Karatsuba method [KO63] for quadratic extensions allows us to compute multiplications in $\mathbb{F}_{q^{2u}}$ using 3 multiplications in $\mathbb{F}_{q^u}$, or to compute a squaring in $\mathbb{F}_{q^{2u}}$ using only 2 multiplications in $\mathbb{F}_{q^u}$. The same method applied to cubic extensions allows us to compute multiplications in $\mathbb{F}_{q^{3u}}$ using only 6 multiplications in $\mathbb{F}_{q^u}$ (rather than 9), and squarings in $\mathbb{F}_{q^{3u}}$ using 6 $\mathbb{F}_{q^u}$-squarings (which are faster than $\mathbb{F}_{q^u}$-multiplications in general). There are also other methods and variations which are competitive for these small extensions, such as the Toom-Cook method [Too63, CA66], which computes an $\mathbb{F}_{q^{3u}}$ multiplication using only 5 $\mathbb{F}_{q^u}$ multiplications, but this requires a substantially higher number of additions. A helpful report that compares all of these methods in the contexts of pairings is given by Devegili *et al.* [DOSD06]. Referring back to the examples

above, and this time descending down the tower using Karatsuba multiplications for the quadratic and cubic extensions gives that $\mathbb{F}_{q^{12}}$ multiplications now cost $3 \cdot 6 \cdot 3 = 54$ $\mathbb{F}_q$ multiplications; a huge improvement over the schoolbook method. We note that a different ordering of the quadratic and cubic towers from $\mathbb{F}_q$ to $\mathbb{F}_{q^{12}}$ could be chosen, and that this would give the same number of $\mathbb{F}_q$ multiplications for a multiplication in $\mathbb{F}_{q^{12}}$, but that there are certainly reasons (other than the twisted curve) that we would prefer one tower over another. We will discuss this in closer detail in Chapter 5.

It could potentially be misleading however, to argue that the low number of $\mathbb{F}_q$ multiplications offered by degree 2 and 3 Karatsuba-like methods is what makes the towered extensions preferable to a direct extension. Indeed, the Karatsuba and Toom-Cook algorithms generalise to extensions of any degree [WP06], [Ber01, §6]. In fact, generalised Toom-Cook theoretically guarantees that we will be able to perform the $\mathbb{F}_{q^{12}}$ multiplication from the above example (via a direct extension) using only 23 $\mathbb{F}_q$ multiplications, which is less than half the number of $\mathbb{F}_q$ multiplications used in our towered Karatsuba approach. However, such high-degree generalisations require an enormous number of $\mathbb{F}_q$ additions, and the theoretical number of multiplications they save is nowhere near enough to offset this deficit. Thus, technically speaking, it is in the saving of $\mathbb{F}_q$-additions that the towered approach gains its advantage. Indeed, the additions encountered when performing the highest level multiplications at the top most sub-extension of the tower filter down linearly to $\mathbb{F}_q$, whilst performing $\mathbb{F}_{q^k}$-arithmetic via a direct extension blows the number of additions out (at the very least) quadratically.

Given the simple test to determine irreducibility of the binomial $x^k - \alpha$ when $q \equiv 1 \bmod 12$ and $k = 2^i 3^j$ above, Koblitz and Menezes defined a *pairing-friendly field* to be a prime field with characteristic q of this form. However, given the number of conditions already imposed on the search for pairing-friendly curves, Benger and Scott argue that this extra restriction is unnecessary [BS10]. They relax this constraint and introduce the notion of *towering-friendly fields*: a field $\mathbb{F}_{q^m}$ is called towering-friendly if all prime divisors of m also divide $q - 1$. For such fields, they invoke Euler's conjectures to give an irreducibility theorem that facilitates all intermediate subextensions to be constructed via a binomial – see Theorem C.1. We make extensive use of this theorem when proving results in Chapter 5 and Chapter 6.

2.6.4 Loop shortening

Loop shortening has played a major role in the evolution of pairing computation. Indeed, the series of landmark works that are summarised in this section have an impressive evolution of their own. Duursma and Lee [DL03] were the first to show that, in special cases, a bilinear pairing can be obtained without iterating Miller's algorithm as far as the large prime group order r. Barreto *et al.* [BGOS07] generalised this observation to introduce the η_T pairing (the *eta pairing*); a pairing which achieves a much shorter loop length (than r) on any supersingular curve. Hess, Smart and Vercauteren [HSV06] simplified and extended the η_T pairing to ordinary curves, introducing the *ate pairing*, whose loop length is $T = t - 1$, where t is the trace of the Frobenius endomorphism (see Eq. (2.6)), which is much smaller than r in general cases of interest. A number of authors followed this work with observations that in many cases we can do even better than the ate pairing. This included the introduction of the *R-ate pairing* [LLP09], as well as other optimised variants of the ate pairing [MKHO07]. Vercauteren [Ver10] culminated all of these works and introduced the notion of *optimal pairings*, conjecturing a lower bound on the loop length required to obtain a bilinear pairing on any given curve, and showing how to achieve it in many cases of interest. His conjecture was proven soon after by Hess, who drew a line under all the loop-shortening work to date, putting forward a general framework that encompasses all elliptic curve pairings [Hes08].

Our intention in this section is to bring the reader up to speed with optimal pairings, by picking a few examples that illustrate key concepts. For the sake of simplicity, we are forced to skip past some of the key works mentioned in the last paragraph; in particular, we will not present the η_T pairing that targets supersingular curves, since it is most suited to curves over fields of characteristic 2 and 3. We will also not be giving examples of the works that came between the ate and optimal ate pairing papers (e.g. [MKHO07, LLP09]), in hope that the reader will not have too much trouble following an immediate generalisation.

At a high level, the notion of loop shortening makes use of two observations. Firstly, recall from Section 2.1 (in particular Example 2.1.23), that appropriate endomorphisms on E compute some multiple $[\lambda]P$ from P, which essentially allow us to "skip ahead" in the fundamental computation of $[m]P$ from P. Just as they can be used to shorten the double-and-add loop for scalar multiplications in ECC, efficient endomorphisms can be used to shorten the Miller loop in PBC. The

second observation is that, given any two bilinear pairings on E, their product or quotient will also give a bilinear pairing. More generally, we can say that if $e_1, ..., e_n$ are bilinear pairings on E, then $\prod_i e_i^{j_i}$ ($j_i \in \mathbb{Z}$) will also be a bilinear pairing [ZZH08a, Corr. 1].

We start with an example of Scott's idea [Sco05b], which came from the first paper to look at loop shortening on any type of ordinary curve. He looked at a special case of ordinary curves called *not supersingular curves* (NSS). These should not be confused with the more general term non-supersingular, which (by definition) means all ordinary curves. NSS curves are a special type of ordinary curve, but they cover the cases that are most useful in the context of pairings. In fact, we have already seen NSS curves, as they are precisely the curves described in Table 2.2. Essentially, the modularity conditions imposed on the curves $y^2 = x^3 + b$ and $y^2 = x^3 + ax$ in Table 2.1 is what makes them supersingular, because these conditions force the maps ϕ described in that table to be defined over the extension field – i.e. these congruences make ϕ a distortion map. On the other hand, the alternative modularities on the same curves in Table 2.2 mean that the associated ϕ's are defined over $\mathbb{F}_q$. Thus, Scott starts with the motivating question: *under these circumstances, what becomes of these distortion maps?* The rest of his paper responds by showing that they are useful, not as distortion maps, but rather as efficient endomorphisms on E. The following example does not give the details of Scott's algorithm; it merely hints towards it by showing the potential of the endomorphisms ϕ on an NSS curve.

Example 2.6.5 (Magma script). Taking $x = -1$ generates the smallest BN curve (see Example 2.5.6 for the polynomials) with $q = 19$, $E/\mathbb{F}_q : y^2 = x^3 + 2$ and $r = 13$ as the group order. It is clearly an NSS curve (see Table 2.2 or [Sco05b, Eq. 4]). The non-trivial cube roots of unity are defined over $\mathbb{F}_q$, and are $\zeta_3 = 7$ and $\zeta_3^2 = 11$. They both define a different endomorphism on E (e.g. $\zeta_3 : (x, y) \mapsto (\zeta_3 x, y)$) which corresponds to a different scalar multiplication λ, i.e. $(\zeta_3 x, y) = [\lambda](x, y)$. The two different λ's are the solutions of $\lambda^2 + \lambda + 1 = 0 \bmod r$, which comes from $\lambda^3 \equiv 1 \bmod r$ matching $\zeta_3^3 = [1]$ in $\mathrm{End}(E)$, so $\lambda_1 = 9$ and $\lambda_2 = 3$ correspond to ζ_3 and ζ_3^2 respectively. Miller's algorithm would usually double-and-add to compute $[r]P = [\lambda^2 + \lambda + 1]P = [\lambda]([\lambda]P + P) + P$. However, for $P = (x, y)$, the endomorphism allows us to easily calculate the point $[\lambda]P + P = (-(\lambda + 1)x, -y)$. Thus, if we store the values of the points in the $n = \lfloor \log_2 \lambda \rfloor$ doublings that build up to $[\lambda]P$, the values of the points

in the second n doublings can be found at the cost of a single multiplication. This is already more efficient, but Scott notices that since the points are related, the lines they contribute in the point doubling phase of Miller's algorithm are similarly related. Namely, the contribution to the pairing value in the first n iterations is $(y_Q - y_i) - m_i(x_Q - x_i)$, where (x_i, y_i) is the point $[2^i]P$, and m_i is the line slope resulting for the point doubling (we use m in this example because λ is already taken). It follows (see [Sco05b, §5]) that the contribution to the pairing value from the final n doublings will be $(-y_Q - y_i) - m_i(\lambda x_Q - x_i)$. This means we only need to loop as far as $n = \lfloor \log_2 \lambda \rfloor$ (rather than $2n = \lfloor \log_2 \lambda^2 \rfloor$) to get all the information we need. See Scott's paper for the algorithm description that ties all this together, where he deals with cases where $\lambda = 2^a + 2^b$. Thus, to finish our example with the algorithm write $\lambda_1 = 2^{a_1} + 2^{b_1}$ and $\lambda_2 = 2^{a_2} + 2^{b_2}$ with $a_1 = 3$, $b_1 = 0$, $a_2 = 1$, $b_2 = 0$.

The ϕ maps on NSS curves clearly offer an advantage, but there is another endomorphism we have already seen that turns out to be much more powerful. Namely, the ate pairing makes use of the Frobenius endomorphism π on E. A key observation is that the Frobenius endomorphism acts trivially on elements in the base field, i.e. $\pi(P) = P$ in $\mathbb{G}_1$, so we instead look at using the trace-zero subgroup $\mathbb{G}_2$ where π acts non-trivially. Here $\pi(Q) = [q](Q)$, but since $[q](Q) = [t-1](Q)$, we have $\pi(Q) = [T](Q)$ (recall that $T = t - 1$). Hess, Smart and Vercauteren [HSV06] use this endomorphism to derive the ate pairing a_T, which is a map

$$a_T : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_T,$$

defined as

$$a_T(Q, P) = f_{T,Q}(P)^{(q^k-1)/r}.$$

It helps to see a brief sketch of their proof as follows. We show that a_T is bilinear by relating its value $f_{T,Q}(P)^{(q^k-1)/r}$ to the Tate pairing (with Q as the first argument), which we already know is bilinear. Since $q \equiv T \bmod r$, $T^k \equiv 1 \bmod r$ (because k is the embedding degree), so write $mr = T^k - 1$ for some m. Recall the Tate pairing (with Q as the first argument) as $e(Q, P) = f_{r,Q}(P)^{(q^k-1)/r}$, which (under simple properties of divisors) means $e(Q, P)^m = f_{mr,Q}(P)^{(q^k-1)/r} = f_{T^k-1,Q}(P)^{(q^k-1)/r}$. We can then (again using simple properties of divisors) split

this into a product of $f_{T,[T^i]Q}(P)$, each of which is raised to an appropriate exponent. Since $Q \in \mathbb{G}_2$, each of these $[T^i]Q$'s is the same as $\pi^i(Q)$, and since π is purely inseparable of degree q, all of the values $f_{T,[T^i]Q}(P)$ in the product become $f_{T,Q}^{q^i}(P)$, so we can clean up the exponent to get $e(Q,P) = a_T(Q,P)^v$. The exponent v does not divide r in general, so the bilinearity of the ate pairing follows from that of the Tate pairing (see [HSV06, Th. 1] for the full details).

Since there is a final exponentiation, the optimisations that transformed Miller's algorithm into the BKLS version still apply, so we only need to update the input definitions in Algorithm 2.2. Namely, r becomes T, P and Q (from $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively) switch roles. For no other reason than for ease of future reference, we write these updates in an ate-specific version below. Note that if $T = t - 1 < 0$, then it is fine to take $T = |T|$ [Ver10, §C]. There is only one trick that was used in the Tate pairing that does not carry across to the ate setting. Namely, we can no longer ignore the last bit in the final iteration like we did in Section 2.6.1, because if an addition occurs in the final iteration it will now be a sloped line, whilst in the Tate pairing the last addition line joined P and $[r-1]P = -P$ and was therefore vertical.

Algorithm 2.3 The BKLS-GHS version of Miller's algorithm for the ate pairing.

Input: $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ (Type 3 pairing) and $T = (T_{n-1} \dots T_1 T_0)_2$ with $T_{n-1} = 1$.
Output: $f_{T,Q}(P)^{(q^k-1)/r} \leftarrow f$.

1: $R \leftarrow Q$, $f \leftarrow 1$.
2: **for** $i = n-2$ down to 0 **do**
3: Compute the sloped line function $\ell_{R,R}$ for doubling R.
4: $R \leftarrow [2]R$.
5: $f \leftarrow f^2 \cdot \ell_{R,R}(P)$.
6: **if** $r_i = 1$ **then**
7: Compute the sloped line function $\ell_{R,Q}$ for adding R and Q.
8: $R \leftarrow R + Q$.
9: $f \leftarrow f \cdot \ell_{R,Q}(P)$.
10: **end if**
11: **end for**
12: **return** $f \leftarrow f^{(q^k-1)/r}$.

Example 2.6.6 (Magma script). It helps to immediately see the difference between the ate and Tate pairing, so we will continue on from Example 2.6.2: $q = 47$, $E/\mathbb{F}_q : y^2 = x^3 + 21x + 15$, $\#E(\mathbb{F}_q) = 51$, $r = 17$, $k = 4$, $\mathbb{F}_{q^4} = \mathbb{F}_q(u)$, $u^4 - 4u^2 + 5 = 0$, $P = (45, 23) \in \mathbb{G}_1$ and $Q = (31u^2 + 29, 35u^3 + 11u) \in \mathbb{G}_2$. The trace of Frobenius is $t = -3$, so take $T = 4$. Thus, we will compute the ate

pairing via Algorithm 2.3 with only two doublings. We have combined the indeterminate function ℓ and its evaluation $\ell(P)$ at P into the same column to fit the table in. Thus, the ate pairing a_T is computed as $a_T(Q,P) = f_{r,Q}(P)^{(q^k-1)/r} =$

i/ R_i	steps of Alg. 2.1	point R	update (ℓ); update at P ($\ell(P)$)	paired value f
	1	$(31u^2+29, 35u^3+11u)$		1
1/0	3-5	$(7u^2+25, 37u^3+28u)$	$y+(u^3+32u)x+42u^3+15u$; $40u^3+45u+23$	$40u^3+45u+23$
0/0	3-5	$(16u^2+12, 6u^3+24u)$	$y+(28u^3+22u)x+17u^3+26u$; $8u^3+29u+23$	$44u^3+24u^2+41u+31$
	12		$f_{r,Q}(P) \leftarrow 44u^3+24u^2+41u+31$	

$(44u^3+24u^2+41u+31)^{287040} = 21u^3+37u^2+25u+25$.

Notice the price we pay for the much shorter loop in the ate pairing, in that it is now the first argument of the pairing (Q) that is defined over the larger field, so the elliptic curve operations (doublings/additions) and line function computations are now taking place in $\mathbb{F}_{q^k}$. For example, compare the second and third columns of the table in Example 2.6.6 to the table in Example 2.6.2. It is here that the power of a high-degree twist really aids our cause. Namely, utilising the twisting isomorphism allows us to move the points in $\mathbb{G}_2$, which is defined over $\mathbb{F}_{q^k}$, to points in $\mathbb{G}_2'$, which is defined over the smaller field $\mathbb{F}_{q^{k/d}}$. In Example 2.6.6 above where $k=4$, the maximum degree twist permitted by E is $d=2$, so we could have performed the point operation and line computations in $\mathbb{F}_{q^{k/2}} = \mathbb{F}_{q^2}$. However, if the curve had have been of the form $y^2 = x^3 + ax$, we could have utilised a $d=4$ quartic twist (see Section 2.3.3) and performed these operations all the way down in the base field $\mathbb{F}_q$; i.e. in this case we would pay no price for a much smaller loop. In general though, provided we make use of high-degree twists in the ate pairing, then the price we pay in doing more work (per iteration) in the larger field is nowhere near enough to offset the savings we gain through having a much shorter loop, meaning that the ate pairing (or one of its variants) is much faster than the Tate pairing. We now turn to describing optimal pairings. Vercauteren [Ver10] begins with the observation that the ate pairing a_T corresponding to $T \equiv q \bmod r$ is a special case of the pairing a_{λ_i} that is obtained by taking any power $\lambda_i \equiv q^i \bmod r$; some specific consequences of this observation were previously considered in [MKHO07, ZZH08b]. Since λ_i corresponds to the loop length of the pairing a_{λ_i}, we would like it to be as small as possible. Thus, we would like to find the smallest value of $q^i \bmod r$ ($i \in \mathbb{Z}$), and since $q^k \equiv 1 \bmod r$, finding the smallest a_{λ_i} would only require

testing the possibilities up to $k-1$ ($i=k$ clearly gives the trivial degenerate pairing). However, Vercauteren actually does much better than this by observing that since $q^i \bmod r$ induces a bilinear pairing a_{λ_i}, then any linear combination of $\sum_{i=0}^{l} c_i q^i \equiv 0 \bmod r$ gives rise to a bilinear pairing

$$(Q,P) \mapsto \left(\prod_{i=0}^{l} f_{c_i,Q}^{q^i}(P) \cdot \prod_{i=0}^{l-1} \ell_i \right)^{(q^k-1)/r}, \tag{2.23}$$

where the ℓ_i are simple "one-off" line functions (chords) that are needed to make the bilinearity hold – see [Ver10, Eq. 7] for details. Also, the exponentiations of each of the (at most $\ell+1$) line functions to the power of q^i should not concern us, as these are just repeated applications of the Frobenius endomorphism in $\mathbb{G}_T$, which is essentially cost-free (more on this in Section 2.6.6). The main point to note is that the loop lengths of the Miller functions $f_{c_i,Q}$ are the c_i. Thus, we would like to find a multiple mr of r with a base-q expansion $mr = \sum_{i=0}^{l} c_i q^i$ that has the smallest c_i coefficients possible. Vercauteren proceeds naturally by posing this search as a lattice problem, i.e. that such small c_i are obtained by solving for short vectors in the following lattice

$$L = \begin{pmatrix} r & 0 & 0 & \dots & 0 \\ -q & 1 & 0 & \dots & 0 \\ -q^2 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \ddots & \\ -q^{\varphi(k)-1} & 0 & \dots & 0 & 1 \end{pmatrix}, \tag{2.24}$$

which is spanned by the rows, and where $\varphi(k)$ is the Euler phi function of k. He then invokes Minkowski's theorem [Min10] to show that there exists a short vector $(v_1, ..., v_{\varphi(k)-1})$ in L such that $\max_i |v_i| \leq r^{1/\varphi(k)}$. Thus, we have an upper bound on the largest Miller loop length that will be encountered when computing the pairing in (2.23). Vercauteren uses this bound to define an optimal pairing [Ver10, Def. 3]: $e(\cdot,\cdot)$ is called an *optimal pairing* if it can be computed in $\log_2 r/\varphi(k) + \epsilon$ Miller iterations, with $\epsilon \leq \log_2 k$. He subsequently conjectures that any bilinear pairing on an elliptic curve requires at least $\log_2 r/\varphi(k)$ Miller iterations. Following [Ver10, Def. 3], Vercauteren also notes that the reason that the dimension of L is $\varphi(k)$ is because we really only need to consider q^i up to $q^{\varphi(k)-1}$. This is due to that fact that $\Phi_k(q) \equiv 0 \bmod r$ implies that q^j

with $j > \varphi(k)$ can be written as linear combinations of the q^i ($i \leq \varphi(k) - 1$) with small coefficients, which means only these q^i should be considered linearly independent.

Before giving examples, we mention a caveat. Observe that $\max_i |c_i| \leq r^{1/\varphi(k)}$ does not imply that the lower bound is met, since the number of Miller iterations required is given by $\sum_i \log_2 c_i$. However, we will be searching for small vectors in the lattice L, where q and r come from families and are therefore given as polynomials $q(x)$ and $r(x)$. Therefore, the c_i in the short vectors will themselves be polynomial expressions $c_i(x)$, meaning that the Miller functions $f_{c_i(x),Q}$ in (2.23) will typically follow from $f_{x,Q}$.

We will illustrate with three families that were used as examples in Section 2.5. Vercauteren gives more examples. Magma has a built in algorithm `ShortestVectors`() that serves our purpose, but the code we use in the following three examples was written by Paulo Barreto, and passed on to us by Luis Dominguez Perez.

Example 2.6.7 (Magma script). Recall the parameterisations for $k = 12$ BN curves from Example 2.5.6: $t(x) = 6x^2 + 1$, $q(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$ and $r(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$. These were actually used to generate the curve in Example 2.5.2, with $x = 9453956337761452438$ being 67 bits, which generated a 271-bit q and r. Observe that Miller's algorithm to compute $f_{r,P}(Q)$ in the Tate pairing would therefore require around 270 iterations. Alternatively, $t = t(x)$ is 137 bits, so computing the ate pairing $a_T(Q, P) = f_{T,Q}(P)^{(q^k-1)/r}$ would require around 136 iterations. However, Vercauteren's bound suggests we can do even better: since $\varphi(12) = 4$, our loop can be reduced by a factor of 4, i.e. we should require $\log_2 r/4 \approx 68$ iterations. Following (2.24) then, we seek short vectors in the lattice

$$L = \begin{pmatrix} 36x^4 + 36x^3 + 18x^2 + 6x + 1 & 0 & 0 & 0 \\ -6x^2 & 1 & 0 & 0 \\ 36x^3 + 18x^2 + 6x + 1 & 0 & 1 & 0 \\ 36x^3 + 24x^2 + 12x + 3 & 0 & 0 & 1 \end{pmatrix},$$

where the $-q(x)^i$ down the first column were immediately reduced modulo $r(x)$. Some short vectors in L are $V_1(x) = (6x+2, 1, -1, 1)$, $V_2(x) = (6x+1, 6x+3, 1, 0)$, $V_3(x) = (-5x - 1, -3x - 2, x, 0)$, $V_4(x) = (2x, x + 1, -x, x)$. In reference to the point we made before this example, we prefer the short vectors with the minimum number of coefficients of size x, so choosing $V_1(x)$ and computing the optimal

ate pairing $a_{V_1(x)}$ following (2.23) gives

$$\begin{aligned} a_{V_1(x)} &= (f_{6x+2,Q}(P) \cdot f_{1,Q}(P) \cdot f_{-1,Q}(P) \cdot f_{1,Q}(P) \cdot M)^{(q^k-1)/r}, \\ &= (f_{6x+2,Q}(P) \cdot M)^{(q^k-1)/r}, \end{aligned}$$

where $f_{1,Q} = 1$ and $f_{-1,Q} = 1/f_{1,Q}v_Q$ (which disappears in the final exponentation) can be discarded, and M is a product of 3 simple line functions that are computed easily – this example is in [Ver10, IV.A], where M is defined. The only Miller loop we need to compute is $f_{6x+2,Q}(P)$, which for our x-value, is 69 bits, meaning the optimal pairing indeed requires $\log_2 r/4 \approx 68$ iterations. Notice then, the difference between the ease of using $V_1(x)$ compared to any of the other short vectors above, which all suggest more than one Miller loop.

Example 2.6.8 (Magma script). Recall the parameterisations for $k = 16$ KSS curves from Example 2.5.7 as $t(x) = (2x^5 + 41x + 35)/35$, $q(x) = (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980$ and $r(x) = (x^8 + 48x^4 + 625)/61250$. For any x-value, the Tate pairing requires computing the function $f_{x^8+48x^4+625,P}(Q)$, whilst the ate pairing computes the function $f_{(2x^5+41x+35)/35,Q}(P)$. Since $\varphi(k) = 8$, the ate pairing is not optimal, i.e. $\log_2 r/8$ should have an optimal pairing loop length of order $O(x)$, not $O(x^5)$. Thus, we look for short vectors in the lattice

$$L = \begin{pmatrix} x^8+48x^4+625 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2x^5-41x & 35 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4x^6+117x^2 & 0 & 175 & 0 & 0 & 0 & 0 & 0 \\ 2x^7-29x^3 & 0 & 0 & 875 & 0 & 0 & 0 & 0 \\ 1x^4+24 & 0 & 0 & 0 & 7 & 0 & 0 & 0 \\ -1x^5-38x & 0 & 0 & 0 & 0 & 35 & 0 & 0 \\ -3x^6-44x^2 & 0 & 0 & 0 & 0 & 0 & 175 & 0 \\ 11x^7+278x^3 & 0 & 0 & 0 & 0 & 0 & 0 & 875 \end{pmatrix}.$$

A nice short vector is $V(x) = (x, 1, 0, 0, 0, -2, 0, 0)$, so indeed an optimal pairing is

$$a_{V(x)} = (f_{x,Q}(P) \cdot f_{-2,Q}(P) \cdot M)^{(q^k-1)/r},$$

where M is again a product of simple one-off lines, and we can compute $f_{-2,Q}(P)$ as $1/f_{2,Q}(P)$, since the vertical line that makes two equal evaporates in the final exponentiation. Note that $f_{2,Q}(P)$ is simply the first doubling of Q at P, and

that $f_{x,Q}(P)$ is the only Miller loop required.

Example 2.6.9 (Magma script). Recall the parameterisations for a $k = 24$ BLS curve from Example 2.5.5 as $t(x) = x + 1$, $q(x) = (x-1)^2(x^8 - x^4 + 1)/3 + x$ and $r(x) = \Phi_{24}(x) = x^8 - x^4 + 1$. The Tate pairing requires the computation $f_{x^8-x^4+1,P}(Q)$ whilst the ate pairing computes $f_{x,Q}(P)$. Since $\varphi(k) = 8$, the ate pairing is already optimal, i.e. it has a loop length of $\log_2(r)/8$. In cases when the ate pairing is not optimal, like the previous two examples, it is common that other variants like the R-ate pairing of [LLP09] also achieve optimality. For example, Scott uses the R-ate pairing to achieve optimality for $k = 12$ and $k = 18$ implementations targeting the 128 and 192-bit security levels [Sco11, Table 1].

2.6.5 Low Hamming weight loops

This very short subsection describes a more obvious optimisation to Miller's algorithm. This trick was suggested in the very early papers on pairing computation, but for reasons that will become clear in a moment, we have delayed its introduction in this section until after we described the ate and optimal ate pairings. Regardless of the pairing-based protocol, the loop length of the pairing is known publicly; therefore, unlike ECC where we try to avoid special choices of scalars that might give attackers unnecessary advantage, in PBC there is no problem in specialising the choice of the loop length. In this light, it is advantageous to use curves where the loop length has a low Hamming weight, thus minimising the number of additions incurred in Miller's algorithm.

For supersingular curves over prime fields, where $\#E(\mathbb{F}_q) = q + 1$, finding a curve whose large prime divisor r has low Hamming weight is relatively easy. Thus, in the early days, facilitating a low Hamming weight Miller loop was not too difficult. However, once the introduction of parameterised families were needed for higher embedding degrees, the polynomial representation for $r(x)$ meant that controlling the loop length (r) of the Tate pairing was a little more difficult. The best we could do in this scenario is search for x values of low Hamming weight, in the hope that the polynomial $r(x)$ wouldn't completely destroy this. Nowadays however, the introduction of the ate and optimal ate pairings makes this optimisation very relevant. Namely, as we saw in the examples in the previous subsection, the loop length associated with the optimal Miller function is often some small function of x, if not x itself. Thus, choosing x to be of low Hamming

weight can be very advantageous for a faster Miller loop, as we show in the following example. In fact, we will see in the next subsection that a faster Miller loop is only a partial consequence.

Example 2.6.10 (Magma script). Both $x = 258419657403767392$ and $x = 144115188109674496$ are 58-bit values that result in $k = 24$ BLS curves suitable for pairings at the 224-bit security level. The former was found by kick-starting the search at a random value between 2^{57} and 2^{58}, and as such, has a Hamming weight of 28, as we would expect. On the other hand, the second value is actually $2^{57}+2^{25}+2^{18}+2^{11}$, which has Hamming weight 4. Thus, we would much prefer the second value since this would result in 24 less additions through the Miller loop. Another nice alternative that gives similar parameter sizes is $x = 2^{56} + 2^{40} - 2^{20}$, which does not have a low Hamming weight, but rather a low NAF-weight (weight in the signed binary representation), for which Miller's algorithm can be easily updated to take advantage of.

2.6.6 The final exponentiation

Until now, our optimisations have all applied to the Miller loop. This was a natural place to look for tricks and shortcuts in the early days, since at low levels of security, the Miller loop is by far the bottle-neck of the algorithm. However, as the security level increases, the relative cost of the final exponentiation also increases [DS10]. It appears that, all known high-level optimisations considered, pairings on BN curves at the 128-bit security level is roughly the "crossover point" where the complexities of the Miller loop and the final exponentiation are similar [AKL+11, Table 4], [BGDM+10, Table 3], [NNS10, Table 2]. Thus, at higher levels of security, the final exponentiation is the most time-consuming stage of the pairing computation.

For curves belonging to families, Scott *et al.*'s algorithm [SBC+09a] is the fastest method to date. In this subsection, we illustrate their technique by means of an example, which we take directly from our joint work with Kristin Lauter and Michael Naehrig [CLN11]. This work looked at $k = 24$ BLS curves in detail, since this family is a frontrunner for high-security pairings, particularly when targeting 256-bit security. There are several other examples looked at in [SBC+09a].

We start with a brief description of the general algorithm, before applying it to our particular case. Suppose k is even and write $d = k/2$. We start by

splitting the final exponent into three components

$$(q^k - 1)/r = \underbrace{[(q^d - 1)] \cdot [(q^d + 1)/\Phi_k(q)]}_{\text{easy part}} \cdot \underbrace{[\Phi_k(q)/r]}_{\text{hard part}},$$

where the two components on the left are the "easy part" because (the second bracket reduces to powers of q and) raising elements in $\mathbb{F}_{q^k}$ to the power of q involves a simple application of the Frobenius operator π, which almost comes for free. It is the $\Phi_k(q)/r$ term that does not reduce to such a form and which is aptly named the "hard part" of the final exponentiation. Suppose we have already exponentiated through the easy part, and our intermediate value is $m \in \mathbb{F}_{q^k}$. The straightforward way to perform the hard part, i.e. $m^{\Phi_k(q)/r}$, is to write the exponent in base q as $\Phi_k(q)/r = \sum_{i=0}^{n-1} \lambda_i q^i$, and to further exploit repeated applications of π in

$$m^{\Phi_k(q)/r} = (m^{q^{n-1}})^{\lambda_{n-1}} \dots (m^q)^{\lambda_1} \cdot m^{\lambda_0},$$

so that all the m^{q^i} terms essentially come for free, and the hard part becomes the individual exponentiations to the power of the λ_i, which are performed using generic methods. These methods, however, do not take advantage of the polynomial description of q, which is where Scott *et al.*'s work advances beyond the more obvious speed-ups.

Example 2.6.11 (Magma script). Recall the $k = 24$ BLS parameterisations from Example 2.6.9: $t(x) = x + 1$, $q(x) = (x - 1)^2(x^8 - x^4 + 1)/3 + x$ and $r(x) = \Phi_{24}(x) = x^8 - x^4 + 1$. To give an idea of the task we are up against, suppose we are targeting the 256-bit security level, as we did with these curves in Example 2.5.5 with $x = 9223372036854782449$. The final exponentiation in this case involves raising a 15082-bit value $f \in \mathbb{F}_{q^{24}}$, to the 14578-bit exponent $(q^{24} - 1)/r$, a number far bigger than what we would like to write here (but see the corresponding script). Performing this exponentiation using a naive square-and-multiply with no optimisations would therefore involve 14578 squarings and roughly half as many multiplications in the 15082-bit field, a computation that would blow out the pairing complexity by several orders of magnitude. To take a much faster route, we start by splitting the exponent as

$$(q^{24} - 1)/r = \underbrace{[(q^{12} - 1) \cdot (q^4 + 1)]}_{\text{easy part}} \cdot \underbrace{[(q^8 - q^4 + 1)/r]}_{\text{hard part}}.$$

We compute $f^{(q^k-1)/r} = \left(f^{(q^{12}-1)\cdot(q^4+1)}\right)^{(q^8-q^4+1)/r}$. The exponentiation inside the parentheses is almost free, since $f^{q^{12}}$ is just 12 repeated applications of the Frobenius operation π, and similarly for raising to the power of q^4, so the easy part essentially incurs just a couple of multiplications and maybe an inversion. We are now left with the exponent $(q^8 - q^4 + 1)/r$, for which we can not pull out any more "easy factors". However, a very helpful observation which aids the remaining computations is that, after the first exponentiation to the power $q^{12}-1$, the value $m \in \mathbb{F}_{q^{24}}$ is now such that its norm is $N_{\mathbb{F}_{q^{24}}/\mathbb{F}_{q^{12}}}(m) = 1$. This allows any inversions in $\mathbb{F}_{q^{24}}$ to be computed for free using a simple conjugation [SB04, NBS08, SBC+09a], and any squarings in $\mathbb{F}_{q^{24}}$ to be computed more efficiently than standard $\mathbb{F}_{q^k}$ squarings [GS10, Kar10, AKL+11]. We now make use of the non-trivial part of the algorithm in [SBC+09a], and write the hard part as

$$(q(x)^8 - q(x)^4 + 1)/r(x) = \sum_{i=0}^{7} \lambda_i(x) q(x)^i.$$

In an appendix of her thesis, Benger [Ben10] computed the λ_i for a range of curve families, including BLS curves with $k = 24$, giving $\lambda_i = \nu_i/3$, where

$$\begin{aligned}
\nu_7(x) &= x^2 - 2x + 1, \\
\nu_6(x) &= x^3 - 2x^2 + x = x \cdot \nu_7(x), \\
\nu_5(x) &= x^4 - 2x^3 + x^2 = x \cdot \nu_6(x), \\
\nu_4(x) &= x^5 - 2x^4 + x^3 = x \cdot \nu_5(x), \\
\nu_3(x) &= x^6 - 2x^5 + x^4 - x^2 + 2x - 1 = x \cdot \nu_4(x) - \nu_7(x), \\
\nu_2(x) &= x^7 - 2x^6 + x^5 - x^3 + 2x^2 - x = x \cdot \nu_3(x), \\
\nu_1(x) &= x^8 - 2x^7 + x^6 - x^4 + 2x^3 - x^2 = x \cdot \nu_2(x), \\
\nu_0(x) &= x^9 - 2x^8 + x^7 - x^5 + 2x^4 - x^3 + 3 = x \cdot \nu_1(x) + 3.
\end{aligned}$$

This representation reveals another nice property exhibited by $k = 24$ BLS curves: namely, a very convenient way to compute the ν_i with essentially just multiplications by x. Letting $\mu_i = m^{\nu_i(x)}$, this structure allows us to write the hard part of the final exponentiation as

$$m^{(q^8-q^4+1)/r} = \mu_0 \cdot \mu_1^p \cdot \mu_2^{p^2} \cdot \mu_3^{p^3} \cdot \mu_4^{p^4} \cdot \mu_5^{p^5} \cdot \mu_6^{p^6} \cdot \mu_7^{p^7},$$

FinalExp	Input: $f_{r,Q}(P) \in \mathbb{F}_{q^{24}}$ and loop parameter x
Initialize $f \leftarrow f_{r,Q}(P)$,	
$t_0 \leftarrow 1/f$, $m \leftarrow \overline{f}$, $m \leftarrow m \cdot t_0, t_0 \leftarrow \pi_q^4(m), m \leftarrow m \cdot t_0$,	
$m_1 \leftarrow m^x$, $m_2 \leftarrow m_1^x$ $m_1 \leftarrow m_1^2$, $m_1 \leftarrow \overline{m_1}$, $\mu_7 \leftarrow m_2 \cdot m_1, \mu_7 \leftarrow \mu_7 \cdot m$,	
$\mu_6 \leftarrow \mu_7^x$, $\mu_5 \leftarrow \mu_6^x$, $\mu_4 \leftarrow \mu_5^x$, $\mu_7' \leftarrow \overline{\mu_7}$, $\mu_3 \leftarrow \mu_4^x$, $\mu_3 \leftarrow \mu_3 \cdot \mu_7'$,	
$\mu_2 \leftarrow \mu_3^x$, $\mu_1 \leftarrow \mu_2^x$, $\mu_0 \leftarrow \mu_1^x$, $m' \leftarrow m^2$, $\mu_0 \leftarrow \mu_0 \cdot m'$, $\mu_0 \leftarrow \mu_0 \cdot m$,	
$f \leftarrow \pi_q(\mu_7), f \leftarrow f \cdot \mu_6$, $f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_5$, $f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_4$,	
$f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_3$, $f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_2$, $f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_1$,	
$f \leftarrow \pi_q(f)$, $f \leftarrow f \cdot \mu_0$,	
Return $f_{r,Q}(P)^{(q^{24}-1)/r} \leftarrow f$.	
	Output: $f_{r,Q}(P)^{(q^{24}-1)/r}$

Table 2.3: The final exponentiation for BLS curves with $k = 24$.

where the μ_i can be computed using the following sequence of operations:

$$\begin{aligned} \mu_7 &= (m^x)^x \cdot (m^x)^{-2} \cdot m,\ \mu_6 = (\mu_7)^x,\ \mu_5 = (\mu_6)^x,\ \mu_4 = (\mu_5)^x, \\ \mu_3 &= (\mu_4)^x \cdot (\mu_7)^{-1},\ \mu_2 = (\mu_3)^x,\ \mu_1 = (\mu_2)^x, \mu_0 = (\mu_1)^x \cdot m^2 \cdot m. \end{aligned}$$

The computation of $m^{(q^8-q^4+1)/r}$ requires 9 exponentiations by x, 12 multiplications in $\mathbb{F}_{q^{24}}$, 2 special squarings, 2 conjugations to compute the inverses and 7 q-power Frobenius operations. We detail a possible scheduling for the full exponentiation routine in Table 2.3. Note that we can simply forget about the difference between the λ_i and the ν_i; by leaving out the 3 in the denominators, we just compute the third power of the pairing.

2.6.7 Other optimisations

There are hundreds of papers that have helped accelerate pairing computation to the point it is at today. Of course, we could not delve into the details of all the optimisations and improvements that are available. For example, since our exposition is largely concerned with computational efficiency, we have not covered the work on compressed pairings [SB04,NBS08,Nae09] which targets low bandwidth environments, or the work by Galbraith and Lin [GL09] which looks at computing pairings using x-coordinates only.

In addition, a number of papers have looked at operations in a pairing-

based protocol that are not the pairing computation itself, the most important of which are point multiplications in the pairing-specific groups $\mathbb{G}_1$ and $\mathbb{G}_2$. In Section 2.5.3 (and Table 2.2 in particular) we saw that the pairing-friendly curves that are most useful in practice are those of the form $E : y^2 = x^3 + b$ or $E : y^2 = x^3 + ax$. In both of these cases there is a non-trivial endomorphism $\phi \in \mathrm{End}(E)$ that facilitates faster point multiplications via GLV/GLS scalar decompositions (refer to Example 2.1.23). For point multiplications in $\mathbb{G}_1$ that take place over the base field, the standard GLV decomposition can make use of ϕ to decompose the scalar. For the more expensive point multiplications in $\mathbb{G}_2$ that take place over extension fields, the GLS technique (which additionally exploits the non-trivial action of the Frobenius endomorphism π) can be used for higher dimensional decompositions. We particularly make mention of the work of Scott *et al.* [SBC+09b] and Fuentes-Castaeda *et al.* [FCKRH11], who consider fast hashing to the group $\mathbb{G}_2$, the bottleneck of which is the expensive cofactor scalar multiplication in $\mathbb{G}_2$. For pairings to become widespread in the industry, efficient off-the-shelf solutions to all the operations involved in pairing-based protocols need to be available.

Finally, we mention that some recent work has revived the potential of the Weil pairing in practice [AKMRH11, AFCK+12]. Indeed, since the complexity of the final exponentiation in the Tate pairing (and its ate-like variants) overtakes that of the Miller loop at higher security levels, it is natural to reconsider the Weil pairing for these scenarios. Although several of the Tate-specific optimisations do not translate across, loop shortening is available in the Weil pairing. Indeed, Hess presented a general framework for loop shortening in both the Tate and Weil pairing methodologies [Hes08]. Aranha *et al.* used this idea to derive Weil pairing variants that are particularly suited to the parallel environment [AKMRH11], and actually showed that their new Weil pairing is substantially faster than the optimal ate pairing when 8 cores are used in parallel.

2.7 Chapter summary

The fundamental computation in ECC is the scalar multiplication which, in the most straightforward case, computes $[m]P$ from $m \in \mathbb{Z}$ and $P \in E$ via a double-and-add routine. Computing the Miller loop in the Tate pairing $e(P, Q)$ can be thought of as an extension of this computation by stipulating that the line

functions used in the scalar multiplication of P are evaluated at Q and accumulated as we proceed to compute $[m]P$. Thus, those who understand ECC related computations should find a relatively easy transition to the basics of pairing computation. This is why we started the chapter with a general overview of ECC in Section 2.1, which included an elementary description of the group law, as well as many optimisations like that of adopting projective coordinates or the GLV technique which exploits endomorphisms to accelerate the computation of $[m]P$. Carrying many ECC related improvements over to the context of PBC is straightforward, whilst translating other optimisations requires a firm knowledge of the functions involved in the pairing computation. For example, one could not hope to thoroughly understand how or why the (optimal) ate pairing works without knowing the basics of divisor theory. In Section 2.2 we presented all the divisor theory that is necessary in preparation for the description of the Weil, Tate and ate-like pairings. We gave a very detailed description of the r-torsion group on E in Section 2.3, and illustrated that the availability of different (efficiently computable) maps between order r subgroups give rise to different pairing types. We adopted the widely accepted argument that Type 3 pairings are most commonly the preferred setting, thereby defining $\mathbb{G}_1$ and $\mathbb{G}_2$ as the base field subgroup and trace-zero subgroup respectively. We finished that section by detailing an efficient method of working in $\mathbb{G}_2$, namely by exploiting the isomorphism between the trace-zero subgroup $\mathbb{G}_2$ on E and the trace-zero subgroup $\mathbb{G}_2'$ on the twisted curve E', which is defined over a smaller field. In Section 2.4 we defined the Weil and Tate pairings and described Miller's algorithm which makes cryptographic pairing computations practical. Having described an efficient algorithm to compute pairings, Section 2.5 looked at the complementary arena of generating pairing-friendly curves. We discussed that pairing-friendly curves are very special in general, and cannot be found by searching at random, before giving a general overview of the many clever methods that have been developed in the last decade to facilitate their construction. We finished in Section 2.6 by bringing the reader up to speed with some of the major milestones in efficient pairing computation, most notably the BKLS-GHS algorithm for the Tate pairing, and the impressive work on loop shortened versions of the Tate pairing which was pinnacled by the optimal ate pairing.

The following chapter presents the first of our novel contributions to the field of pairing computation, which is the optimisation of explicit projective formulas

for the elliptic curve point operations and Miller line computations. Whilst turning affine pairing formulas into projective formulas is essentially straightforward, finding the fastest route to computing them is not so trivial. This involves searching many different elliptic curve models and a number of options for projective coordinate systems, each combination of which requires individual treatment in order to simplify the expressions as much as possible. Chapter 3 presents our record holding explicit formulas for the all of the major scenarios of elliptic curve pairings where projective coordinates perform fastest in practice.

Chapter 3

Fast Miller Functions

This chapter focusses on optimising the explicit formulas for the elliptic curve operations and line computations that take place inside the Miller loop. We present the fastest known explicit formulas for pairings in all practical scenarios where projective formulas perform best; this includes all security levels up to (at least) the 192-bit security level [AKL+11, AFCK+12].

The ultimate goal of this work is best described by referring back to our description of (the BKLS-GHS version of) Miller's algorithm, particularly in the context of the ate pairing. For the doublings (resp. additions) that occur in Algorithm 2.3 then, our aim is to compute Steps 3 and 4 (resp. Steps 7 and 8) as fast as possible. We do not consider optimising the function updates here, i.e. $f \leftarrow f^2 \cdot \ell_{R,R}(P)$ and $f \leftarrow f \cdot \ell_{R,Q}(P)$ in Steps 5 and 9, since this is a separate issue. However, our analysis will include the evaluation of the functions at P, which *is* included in these steps in Algorithm 2.3. Thus, our aim is to optimise the following operations.

– Compute $\ell_{R,R}$ for doubling R.	doubling step
Compute $R \leftarrow [2]R$	(DBL)
Evaluate $\ell_{R,R}(P)$	Steps 3 and 4 of Alg. 2.3
– Compute $\ell_{R,Q}$ for adding R and Q.	addition step
Compute $R \leftarrow R + Q$	(ADD)
Evaluate $\ell_{R,Q}(P)$	Steps 7 and 8 of Alg. 2.3

In the ate pairing variants, the functions $\ell_{R,R}$ and $\ell_{R,Q}$ have coefficients in $\mathbb{F}_{q^k}$,

and they are evaluated at the coordinates of P, which lie in $\mathbb{F}_q$. In the Tate pairing, P and Q switch roles, so the functions now have coefficients in $\mathbb{F}_q$, and are evaluated at the coordinates of Q, which are in $\mathbb{F}_{q^k}$. In the ate pairing then, Q plays a much larger part in the Miller loop computations, and since Q comes from the larger field, optimising the computations in the above steps has a more significant impact on the overall runtime of the pairing algorithm.

3.1 Computing the ate pairing entirely on the twisted curve

Several authors have presented new formulas that achieve faster iterations of the Miller loop on certain curves [CSB04,DS08,IJ08,ALNR10]. The operation counts presented in these papers are given in the context of Tate pairing computations on curves with even embedding degrees, where all elliptic curve operations occur in the base field $\mathbb{F}_q$, and the functions in the Miller loop are evaluated at a point which has one coordinate in $\mathbb{F}_{q^{k/2}}$ and one in the full extension field $\mathbb{F}_{q^k}$. This allows for a relatively simple exposition. However, the ate pairing reverses the roles of the points involved and employs twisted curves. This means that some of the optimisations cannot be applied in the same fashion. The purpose of this section is to tidy up this discussion and to show how operation counts for the Tate pairing can be easily modified to give the analogous ate pairing count.

The usual practice when computing the ate pairing $a_T(Q, P)$ of the points $P \in E(\mathbb{F}_q)$ and $Q \in E(\mathbb{F}_{q^k})$ is to map the point Q to the twisted curve using the isomorphism Ψ^{-1}, so that the point operations (doubling/addition) in the Miller loop can be performed more efficiently using the point $Q' = \Psi^{-1}(Q) \in E'(\mathbb{F}_{q^{k/d}})$, whose coordinates are defined over the smaller field $\mathbb{F}_{q^{k/d}}$. When it is time to compute the Miller line, Q' is "untwisted" back to the full extension field via $Q = \Psi(Q')$ – this is to ensure the pairing value is in $\mathbb{F}_{q^k}$. Operation counts for the Tate pairing do not carry over directly to the ate pairing. In particular, for the Tate pairing it is the y-coordinate of the second argument that is in the full extension field $\mathbb{F}_{q^k}$, whereas one of the coordinates of the first argument in the ate pairing is in $\mathbb{F}_{q^k}$. This means that all optimisations that were based on eliminating subfield elements have to be revised.

Furthermore, pairings on special curves such as Edwards curves and the curves in [CHB+09] impose conditions on cofactors of the group order. Gal-

braith [Gal09] pointed out to us that for twists of degree larger than 2, E and E' can not both simultaneously be in Edwards form. His arguments also apply to the curves given in our original work with sextic twists [CHB+09]. So far this meant that the formulas used for the point operations (on E') and the formulas derived (on E) for the Miller functions must be treated separately which usually results in a greater overall operation count.

In the following theorem, we show that a small (≤ 6) power of the ate pairing can be computed entirely on the twisted curve, rendering the above concerns obsolete. Our pairing can make use of loop shortening techniques just like the ate pairing, but only requires one curve (the twisted curve) to have particular properties. Furthermore, referring back to the individual twist descriptions in Section 2.3.3 shows that most coordinates of the twisted points P' and Q' are defined over subfields. Note that the computation of a small power of pairings for efficiency reasons has been addressed in previous work, see for example [ELM04].

Theorem 3.1. *Let $E/\mathbb{F}_q : y^2 = x^3+ax+b$ and let $E'/\mathbb{F}_{q^{k/d}} : y^2 = x^3+a\omega^4x+b\omega^6$, a degree-$d$ twist of E. Let Ψ be the associated twist isomorphism $\Psi : E' \to E : (x', y') \to (x'/\omega^2, y'/\omega^3)$. Let $P \in G_1$, $Q \in G_2$, and let $Q' = \Psi^{-1}(Q)$ and $P' = \Psi^{-1}(P)$. Let $a_T(Q, P)$ be the ate pairing of Q and P. Then*

$$a_T(Q, P)^{\gcd(d,6)} = a_T(Q', P')^{\gcd(d,6)},$$

where $a_T(Q', P') = f_{T,Q'}(P')^{(q^k-1)/r}$ uses the same loop parameter as $a_T(Q, P)$ on E, but takes the two twisted points Q' and P' as inputs, instead of Q and P.

Proof. Since all factors of the Miller values that lie in a proper subfield of $\mathbb{F}_{q^k}$ vanish under the final exponentiation, it suffices to show that the Miller function updates at each iteration are equal, up to a constant defined over any proper subfield of $\mathbb{F}_{q^k}$. The computation of $a_T(Q, P)$ is composed of addition and doubling steps. Consider the gradients of the lines at either the addition (of $R \leftarrow R + Q$) or the doubling ($R \leftarrow [2]R$) stage of the Miller loop respectively. Let R' and Q' be the twists of R and Q under Ψ^{-1} respectively. We have

$$\frac{y'_R - y'_Q}{x'_R - x'_Q} = \frac{\omega^3(y_R - y_Q)}{\omega^2(x_R - x_Q)} = \omega\frac{y_R - y_Q}{x_R - x_Q} \text{ and } \frac{3x'^2_R + a\omega^4}{2y'_R} = \frac{\omega^4(3x_R^2 + a)}{2\omega^3 y_R} = \omega\frac{3x_R^2 + a}{2y_R}$$

for addition and doubling. We write the update to the Miller function at the

doubling step, $\ell_{R',R'}(P')/v_{[2]R'}(P')$, as

$$\begin{aligned}(\ell_{R',R'}(P'))/(v_{[2]R'}(P')) &= (y_{R'} - y_{P'} - \lambda'(x_{R'} - x_{P'}))/(x_{R'} - x_{[2]U'}) \\ &= (\omega^3 y_R - \omega^3 y_P - \omega\lambda(\omega^2 x_R - \omega^2 x_P))/(\omega^2 x_P - \omega^2 x_{[2]R}) \\ &= \omega \cdot \ell_{R,R}(P)/v_{[2]R}(P),\end{aligned}$$

where λ and λ' are the gradients determined before, on E and E' respectively. In the case of addition, we similarly have $\ell_{R',Q'}(P')/v_{R'+Q'}(P') = \omega \cdot \ell_{R,Q}(P)/v_{R+Q}(P)$. For twists of degree $d = 2$ and $d = 4$, observe that $\omega^2 = \omega^{\gcd(d,6)}$ is in a subfield of $\mathbb{F}_{q^k}$ and thus vanishes in the final exponentiation. Similarly, for $d = 3$ and $d = 6$, ω^3 and ω^6 are both in subfields of $\mathbb{F}_{q^k}$, so that introducing a factor of 3 and 6 respectively to the exponent of $a_T(Q', P')$ will give an identical result to the computation of the same power of $a_T(Q, P)$. □

Corollary 3.2. *If $a_T(Q, P)$ is bilinear and non-degenerate, then so is $a_T(Q', P')$.*

Remark 3.1.1. Note that for $d = 6$ both ω^2 and ω^3 are in proper subfields of $\mathbb{F}_{q^k}$. Thus their contributions to the denominator and numerator vanish in the final exponentiation, so there is no need to introduce a factor of 6 to the final exponent. That is, for sextic twists it is actually always the case that $a_T(Q, P) = a_T(Q', P')$. If denominator elimination is used for $d = 6$, the values differ by ω^3 which lies in a subfield. For $k = 12$ and BN curves this case was considered by Akane, Nogami, and Morikawa [ANM09] who showed that up to constants from subfields $a_T(Q, P) = a_T(Q', P')$. For the other cases either ω^2 or ω^3 lie in a proper subfield $\mathbb{F}_{q^e}$ of $\mathbb{F}_{q^k}$. If 4 or 9 divides $\prod_{d|k} \Phi_d(q)/(q^e - 1)$, respectively, we obtain $\omega^{(q^k-1)/r} = 1$ and thus automatically $a_T(Q, P) = a_T(Q', P')$. However, in general these conditions are not satisfied, and the extra power of 2 or 3 is needed to obtain the same result.

Computing the ate pairing as $a_T(Q', P')$ and using twists as in Section 2.3.3 implies (for $d < 6$) that the only coordinate that lies in the full extension field $\mathbb{F}_{q^k}$ belongs to the second argument; for $d = 6$ all coordinates are defined over subfields, but their being coprime ensures that the paired value lies in the full extension field. In this sense, the field operations encountered in computing the ate pairing $a_T(Q', P')$ on E' mimic the field operations encountered in computing the Tate pairing $e_r(P, Q)$ on E. Thus, point operation and line computation formulas that work in the Tate pairing can directly be applied to the ate pairing.

Inversions in $\mathbb{F}_{q^k}$ are prohibitively expensive and so we will show for all curve types a way to eliminate denominators. Therefore, at the doubling or addition stage of a Miller iteration the update function is given by a polynomial $f = \sum_{i,j} \ell_{i,j} \cdot x_{P'}^i y_{P'}^j$, where the $\ell_{i,j}$ are functions solely of the intermediate point R (doubling) or of the intermediate point R and the base point Q (addition). We reiterate what was said in the beginning of this chapter. In the Tate pairing computation of $e_r(P,Q)$, the $\ell_{i,j}$ are functions of some multiple of the point $P \in E(\mathbb{F}_q)$ and therefore all calculations required to compute the $\ell_{i,j}$ are performed in the base field $\mathbb{F}_q$. Similarly, in the modified definition of the ate pairing computation of $a_T(Q',P')$, the $\ell_{i,j}$ are functions of some multiple of the point $Q' \in E'(\mathbb{F}_{q^e})$ and therefore all calculations required to compute the $\ell_{i,j}$ in this case are performed in the subfield $\mathbb{F}_{q^e}$. Thus, if the computations of the $\ell_{i,j}$ in an iteration of the Tate pairing require $m\mathbf{m}_1 + s\mathbf{s}_1$, where $\mathbf{m}_1$ and $\mathbf{s}_1$ denote multiplication and squaring in $\mathbb{F}_q$, then the equivalent computations in an iteration of the ate pairing will require $m\mathbf{m}_e + s\mathbf{s}_e$, where $\mathbf{m}_e$ and $\mathbf{s}_e$ denote multiplication and squaring in $\mathbb{F}_{q^e}$; a multiplication by the curve constant a costs $\mathbf{d}_a$.

For even embedding degrees (admitting quadratic, quartic or sextic twists) the function update always simplifies to $\ell = \ell_{1,0}x + \ell_{0,1}y + \ell_{0,0}$, so that we have two extra multiplications required here ($\ell_{1,0}$ by x and $\ell_{0,1}$ by y). In the Tate pairing as well as in the ate pairing each of these multiplications costs $e = k/d$ base field multiplications if field extensions are represented in a suitable way. If k is odd and divisible by three and if the curve admits a cubic twist, the function update requires more terms. For comparison, let there be h_{ADD} non-zero terms (excluding $\ell_{0,0}$) in the addition step and h_{DBL} in the doubling step, each of which costs $e = k/3$ base field multiplications. We summarise the situation for different twists in Table 3.1.

k even	DBL	ADD/ mADD
Tate: $e_r(P,Q)$	$m_1\mathbf{m}_1 + s_1\mathbf{s}_1 + 2e\mathbf{m}_1 + \mathbf{m}_k + \mathbf{s}_k$	$m_2\mathbf{m}_1 + s_2\mathbf{s}_1 + 2e\mathbf{m}_1 + \mathbf{m}_k$
Ate: $a_T(Q',P')$	$m_1\mathbf{m}_e + s_1\mathbf{s}_e + 2e\mathbf{m}_1 + \mathbf{m}_k + \mathbf{s}_k$	$m_2\mathbf{m}_e + s_2\mathbf{s}_e + 2e\mathbf{m}_1 + \mathbf{m}_k$
k odd, $3 \mid k$	DBL	ADD/ mADD
Tate: $e_r(P,Q)$	$m_1\mathbf{m}_1 + s_1\mathbf{s}_1 + h_{\mathrm{DBL}}e\mathbf{m}_1 + \mathbf{m}_k + \mathbf{s}_k$	$m_2\mathbf{m}_1 + s_2\mathbf{s}_1 + h_{\mathrm{ADD}}e\mathbf{m}_1 + \mathbf{m}_k$
Ate: $a_T(Q',P')$	$m_1\mathbf{m}_e + s_1\mathbf{s}_e + h_{\mathrm{DBL}}e\mathbf{m}_1 + \mathbf{m}_k + \mathbf{s}_k$	$m_2\mathbf{m}_e + s_2\mathbf{s}_e + h_{\mathrm{ADD}}e\mathbf{m}_1 + \mathbf{m}_k$

Table 3.1: Converting operation counts for single addition and doubling steps in the Tate pairing $e_r(P,Q)$ and ate pairing $a_T(Q',P')$.

In what follows, whenever we omit the subscripts from the operation costs

and write $\mathbf{m}$ and $\mathbf{s}$, we mean $\mathbf{m}_1$, $\mathbf{s}_1$ for Tate pairing computation and $\mathbf{m}_e$, $\mathbf{s}_e$ for ate pairing computation.

Remark 3.1.2. Note that by Theorem 3.1 the computation of $a_T(Q', P')^{\gcd(d,6)}$ can be done entirely on the twisted curve. This means that, for example, Edwards curves can be employed in the ate setting if we choose the original curve such that the twisted curve can be written in Edwards form. We will see another example in Section 3.5.

Following our discussion in Section 2.5.3 (in particular, Table 2.2), optimal methods of curve construction produce curves that admit high-degree twists, either of the form $y^2 = x^3 + b$ for $d = 3$ and $d = 6$, or $y^2 = x^3 + ax$ for $d = 4$. Thus, in the next three sections, we derive specialised formulas that targets each of these cases.

Notation. In [CLN10], we treated the ate and Tate pairings simultaneously by presenting explicit formulas for the pairing $e(R, S)$, where the first argument R was Q' in the modified ate pairing and P in the Tate pairing, and the second argument S was P' in the ate pairing and Q in the Tate pairing. In this exposition however, we will assume we are computing the (modified) ate pairing $a_T(Q', P')$ and present the explicit formulas accordingly. To make this concrete, replace P, Q and R in Algorithm 2.3 with P', Q' and R', since all computations now take place on the twist. For the remainder of this section we let the intermediate multiple of $Q' = (x_2, y_2)$ be $R' = (x_1, y_1)$, and $P' = (x'_P, y'_P)$. Here we adopt numerical subscripts for the first arguments because this facilitates an easier distinction between the coordinates that contribute to the coefficients of the pairing functions, and the coordinate (of P') where the functions are evaluated. In each section we develop doubling and addition formulas, as well as "mixed addition" formulas which account for the fact that the second argument in the pairing remains fixed (and therefore affine) throughout the Miller loop.

3.2 Pairings on $y^2 = x^3 + ax$ with even embedding degrees

The only curves which admit quartic twists over $\mathbb{F}_q$ are of the form $E : y^2 = x^3 + ax$. In this section we assume that the embedding degree k is even and so (by

the discussion in Section 2.3.3) we can use that the x-coordinates of the second argument are defined over a subfield of $\mathbb{F}_{q^k}$, whilst its y-coordinate is defined minimally over $\mathbb{F}_{q^k}$. In the ate pairing $a_T(Q', P')$, this corresponds to x'_P being defined over a subfield, whilst in the Tate pairing $e(P, Q)$, it corresponds to x_Q lying in a subfield. We will assume the ate pairing in our formula derivation.

Curves of the form $E : y^2 = x^3 + ax$ have not received much attention, even for simple elliptic curve arithmetic, e.g. no special formulas were reported in the EFD [BL07a] before our paper. We present new formulas for addition and doubling in a new coordinate system, which we call "weight-$(1, 2)$ coordinates". The point $(X : Y : Z)$ corresponds to the affine point (x, y), where $x = X/Z$ and $y = Y/Z^2$. The projective curve equation for these weights is $Y^2 = X^3Z + aXZ^3$. Lopez and Dahab [LD98] studied such coordinates in the context of elliptic curves over binary fields but these weights have not been used in the context of curves over odd-characteristic fields.

It is quite remarkable that our doubling formulas are faster than any doubling formulas reported for elliptic curves in the EFD.

We extend the explicit formulas for curve operations to compute the doubling and the addition step on these curves. The resulting pairing computations are also significantly faster than their predecessors.

3.2.1 Doubling formulas

For this curve shape the affine doubling formulas to compute $(x_3, y_3) = [2]R' = [2](x_1, y_1)$ simplify to $x_3 = \lambda^2 - 2x_1,\ y_3 = \lambda(x_1 - x_3) - y_1$, where $\lambda = (3x_1^2 + a)/(2y_1)$. In weight-$(1, 2)$ coordinates the doubling formulas to compute $(X_3 : Y_3 : Z_3) = [2](X_1 : Y_1 : Z_1)$ become

$$X_3 = (X_1^2 - aZ_1^2)^2,\ Y_3 = 2Y_1(X_1^2 - aZ_1^2)((X_1^2 + aZ_1^2)^2 + 4aZ_1^2X_1^2),\ Z_3 = 4Y_1^2.$$

The point doubling needs $1\mathbf{m} + 6\mathbf{s} + 1\mathbf{d}_a$ using the following sequence of operations.

$$\begin{aligned} A &= X_1^2,\ B = Y_1^2,\ C = Z_1^2,\ D = aC,\ X_3 = (A - D)^2, \\ E &= 2(A + D)^2 - X_3,\ F = ((A - D + Y_1)^2 - B - X_3),\ Y_3 = E \cdot F,\ Z_3 = 4B. \end{aligned} \tag{3.1}$$

These formulas are now the fastest doubling formulas reported in the EFD: they are faster by 1 **s**-**m** tradeoff than the previous champion, "dbl-20090311-hwcd" due to Hisil, Wong, Carter, and Dawson [BL07a]. Those formulas are optimized for "Doubling-oriented XXYZZR coordinates for Jacobi quartics" and need $2\mathbf{m} + 5\mathbf{s} + 1\mathbf{d}_a$, where a is some curve constant.

3.2.2 Line computation for doubling

In the doubling step of the pairing computation we need to compute $[2]R'$ and compute the line function at R' before evaluating it at $P' = (x_{P'}, y_{P'})$. The affine formula for the computation of $\ell_{R',R'}(P')$ is given as $\frac{\lambda(X_1/Z_1 - x_{P'}) + y_{P'} - Y_1/Z_1^2}{x_{P'} - (\lambda^2 - 2X_1/Z_1)} = -\frac{2Y_1(-(3X_1^2Z_1 + aZ_1^3)\cdot x_{P'} + (2Y_1Z_1)\cdot y_{P'} + X_1^3 - aZ_1^2X_1)}{-(4Y_1^2Z_1)\cdot x_{P'} + 9X_1^4Z_1 + 6aX_1^2Z_1^3 + a^2Z_1^5 - 8X_1Y_1^2}$. Since any element except for $y_{P'}$ is in a proper subfield of $\mathbb{F}_{q^k}$, we can omit computing the entire denominator and also the multiplication by $-Y_1$. We leave the factor of 2 to obtain an **s**-**m** tradeoff. The simplified line function is

$$\ell'_{R',R'}(P') = -2(3X_1^2Z_1 + aZ_1^3)\cdot x_{P'} + (4Y_1Z_1)\cdot y_{P'} + 2(X_1^3 - aZ_1^2X_1).$$

We write $\ell'_{R',R'}(P')$ as $\ell'_{R',R'}(P') = \ell_{1,0}\cdot x_{P'} + \ell_{0,1}\cdot y_{P'} + \ell_{0,0}$ and compute $\ell_{1,0}$, $\ell_{0,1}$ and $\ell_{0,0}$ as

$$\ell_{1,0} = -2Z_1\cdot(3\cdot A + D),\ \ell_{0,1} = 2((Y_1+Z_1)^2 - B - C),\ \ell_{0,0} = (X_1 + A - D)^2 - X_3 - A,$$

using the values computed in (3.1) at an additional cost of $1\mathbf{m} + 2\mathbf{s}$, so that the total operation count for point doubling with line computation is $2(k/d)\mathbf{m}_1 + 2\mathbf{m} + 8\mathbf{s} + 1\mathbf{d}_a$.

3.2.3 Addition and mixed addition

In affine coordinates, the sum $(x_3, y_3) = R' + Q' = (x_1, y_1) + (x_2, y_2)$ is given by $x_3 = \lambda^2 - x_1 - x_2,\ y_3 = \lambda(x_1 - x_3) - y_1$, where $\lambda = (y_1 - y_2)/(x_1 - x_2)$. In weight-$(1,2)$ coordinates this becomes $(X_3 : Y_3 : Z_3) = (X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2)$

$$\begin{aligned} X_3 &= (Y_1Z_2^2 - Y_2Z_1^2)^2 - (X_1Z_2 + X_2Z_1)T, \\ Y_3 &= ((Y_1Z_2^2 - Y_2Z_1^2)(X_1Z_2T - X_3) - Y_1Z_2^2TU)UZ_1Z_2, \\ Z_3 &= (UZ_1Z_2)^2, \end{aligned}$$

where $T = (X_1Z_2 - X_2Z_1)^2 Z_1 Z_2$ and $U = (X_1Z_2 - X_2Z_1)$. This addition can be computed in $10\mathbf{m} + 7\mathbf{s}$ using

$$A = Z_1^2,\ B = Z_2^2,\ C = (Z_1 + Z_2)^2 - A - B,\ D = X_1 \cdot Z_2,\ E = X_2 \cdot Z_1,$$
$$F = Y_1 \cdot B,\ G = Y_2 \cdot A,\ H = (D - E),\ I = 2(F - G),\ II = I^2,\ J = C \cdot H,$$
$$K = 4J \cdot H,\ X_3 = 2II - (D + E) \cdot K,\ Z_3 = J^2,$$
$$Y_3 = ((J + I)^2 - Z_3 - II) \cdot (D \cdot K - X_3) - F \cdot K^2,\ Z_3 = 2Z_3.$$

For mixed addition, i.e. $Z_2 = 1$, the number of operations reduces to $8\mathbf{m} + 5\mathbf{s}$ omitting computation of B, C, D and F.

3.2.4 Line computation for addition and mixed addition

For affine points R', Q', and P' the line function is given by $\ell_{R',Q'}(P') = \frac{\lambda(x_2 - x_{P'}) + y_{P'} - y_2}{x_{P'} - (\lambda^2 - x_1 - x_2)}$. Again, we can omit the denominator because it is entirely defined over a subfield of $\mathbb{F}_{q^k}$. In weight-$(1, 2)$ coordinates the modified line function becomes $\ell'_{Q',R'}(P') = I \cdot X_2Z_2 - I \cdot x_{P'}Z_2^2 + J \cdot y_{P'}Z_2^2 - J \cdot Y_2$. The values $X_2Z_2, x_{P'}Z_2^2$, and $y_{P'}Z_2^2$ do not change during the computation and can thus be precomputed. For the Tate pairing the cost of one addition step (computation of addition and line function) therefore is $(2k/d)\mathbf{m}_1 + 12\mathbf{m} + 7\mathbf{s}$. If $d = 2$ it is possible to save $1\mathbf{m}$ by computing $I \cdot (X_2Z_2 - x_{P'}Z_2^2)$.

When computing the ate pairing, the multiplications in $I \cdot X_2Z_2 - I \cdot x_{P'}Z_2^2 + J \cdot y_{P'}Z_2^2 - J \cdot Y_2$ cost $1\mathbf{m}$ each, given the shape of $x_{P'}$ and $y_{P'}$. The cost of one addition step (computation of addition and line function) in the ate pairing therefore is $14\mathbf{m} + 7\mathbf{s}$.

For mixed additions ($Z_2 = 1$) this simplifies to $\ell'_{m(R',Q')}(P') = I \cdot X_2 - I \cdot x_{P'} + J \cdot y_{P'} - J \cdot Y_2$, costing $(2k/d)\mathbf{m}_1 + 10\mathbf{m} + 5\mathbf{s}$ for both the ate and the Tate pairing for a complete mixed addition step. For $d = 2$ again $1\mathbf{m}$ can be saved in the Tate pairing.

If Q is reused several times in the Tate pairing it might be worthwhile to precompute $1/Y_2$ for longterm usage. At the beginning of a pairing computation $\tilde{X}_2 = X_2/Y_2, \tilde{x}_{P'} = x_{P'}/Y_2$ and $\tilde{y}_{P'} = y_{P'}/Y_2$ are computed. Since Y_2 lies in $\mathbb{F}_q$, $\ell'_{m(R',Q')}(P')$ can be replaced by

$$\ell'_{m(R',Q')}(P')/Y_2 = I \cdot \tilde{X}_2 - I \cdot \tilde{x}_{P'} + J \cdot \tilde{y}_{P'} - J$$

without changing the pairing value. Note also that $\tilde{x}_{P'}$ and $\tilde{y}_{P'}$ are defined over the same fields as $x_{P'}$ and $y_{P'}$ are. In this case a mixed addition step costs only $(2k/d)\mathbf{m}_1 + 9\mathbf{m} + 5\mathbf{s}$.

If instead P' is reused several times in the ate pairing, similar savings are possible. It is useful to precompute $1/y'_{P'}$ and update the function by

$$\bar{f}_{m(R',Q')}(P')/\bar{y}_{P'} = I \cdot \bar{X}_2 - I \cdot \bar{x}_{P'} + J\omega^3 - J \cdot \bar{Y}_2,$$

where $\bar{X}_2 = X_2/\bar{y}_{P'}$, $\bar{x}_{P'} = x_{P'}/\bar{y}_{P'}$, $\bar{Y}_2 = Y_2/\bar{y}_{P'}$, and $y_{P'} = \bar{y}_{P'}\omega^3$, with $\bar{y}_P \in \mathbb{F}_q$. In this case a mixed addition step costs only $(2k/d)\mathbf{m}_1 + 9\mathbf{m} + 5\mathbf{s}$.

Note that these savings are compatible with the saving for $d = 2$.

Depending on the representation of $\mathbb{F}_{q^k}$ over $\mathbb{F}_{q^{k/2}}$ and $\mathbb{F}_{q^{k/d}}$ it is possible to save operations in the other cases.

3.3 Pairings on $y^2 = x^3 + b$ with even embedding degrees

The only curves which can have sextic twists over $\mathbb{F}_q$ are of the form $E : y^2 = x^3 + b$. In this section we assume that the embedding degree k is even and from Section 2.3.3 we can use that the x-coordinate of P' (in $a_T(Q', P')$) is defined over a subfield of $\mathbb{F}_{q^k}$. Note that if $d = 6$, $y_{P'}$ is also defined over a proper subfield, namely $\mathbb{F}_{q^{k/3}}$. In this case the paired value still ends up in $\mathbb{F}_{q^k}$ since the subfields corresponding to $x_{P'}$ and $y_{P'}$ are coprime. For these curves we obtained the best results in standard homogeneous projective coordinates where the curve equation $y^2 = x^3 + b$ becomes $Y^2Z = X^3 + bZ^3$.

3.3.1 Point doubling and line computation

The affine doubling formulas differ from those in Section 3.2 in the definition of λ. We have $\lambda = 3x_1^2/2y_1$. In projective coordinates and after eliminating powers of X_1^3 via the curve equation, we obtain $(X_3 : Y_3 : Z_3) = [2](X_1 : Y_1 : Z_1)$ as

$$X_3 = 2X_1Y_1(Y_1^2 - 9bZ_1^2),\ Y_3 = Y_1^4 + 18bY_1^2Z_1^2 - 27b^2Z_1^4,\ Z_3 = 8Y_1^3Z_1.$$

We homogenize the affine doubling line using $x_1 = X_1/Z_1$ and $y_1 = Y_1/Z_1$ and get

$$\ell'_{R',R'}(P') = 3X_1^2 \cdot x_{P'} - 2Y_1Z_1 \cdot y_{P'} + 3bZ_1^2 - Y_1^2.$$

We write $\ell'_{R',R'}(P') = \ell_{1,0} \cdot x_{P'} + \ell_{0,1} \cdot y_{P'} + L_{0,0}$ and compute $\ell_{1,0}$, $\ell_{0,1}$, $\ell_{0,0}$ and the point $(X_3 : Y_3 : Z_3)$ using the following sequence of operations.

$$\begin{aligned} A &= X_1^2,\ B = Y_1^2,\ C = Z_1^2,\ D = 3bC,\ E = (X_1 + Y_1)^2 - A - B, \\ F &= (Y_1 + Z_1)^2 - B - C,\ G = 3D,\ X_3 = E \cdot (B - G), \\ Y_3 &= (B + G)^2 - 12D^2,\ Z_3 = 4B \cdot F,\ L_{1,0} = 3A,\ L_{0,1} = -F,\ L_{0,0} = D - B. \end{aligned}$$

The total count for the above sequence of operations is $2\mathbf{m} + 7\mathbf{s} + 1\mathbf{d}_b$ in addition to the multiplications by $x_{P'}$ and $y_{P'}$. Note that doubling outside the context of pairings would omit the computation of A and would obtain $E = 2X_1Y_1$, needing a total of $3\mathbf{m} + 5\mathbf{s} + 1\mathbf{d}_b$. As doubling formulas they are not competitive with those in the EFD but they are almost the fastest for the doubling step in pairings, second only to our formulas for the more special curve $y^2 = x^3 + c^2$ in Section 3.5.

3.3.2 Addition, mixed addition and line computation

For the addition of points on $y^2 = x^3 + b$, we adopt the upcoming formulas that were first obtained for curves of the form $y^2 = x^3 + c^2$ in Section 3.5.2. These addition and line computation formulas are independent of the quadratic reciprocity of b in $\mathbb{F}_q$.

3.4 Fast formulas for pairing computations with cubic twists

For an odd embedding degree k, the only possible non-trivial twists are cubic twists and these only exist for curves of the form $y^2 = x^3 + b$, requiring also that $3|k$. Section 2.3.3 showed that in this scenario the point $P' = (x'_P, y'_P)$ has $x_{P'}$ defined over the full extension field $\mathbb{F}_{q^k}$ and $y_{P'}$ defined over a subfield. The formulas obtained in most publications including the previous sections use denominator elimination based on $x_{P'}$ being in a subfield.

In this section we present fast formulas for addition and doubling steps for $y^2 = x^3 + b$ and optimise them using the fact that $y_{P'}, y_{R'}$ and $x_{R'}$ are in a proper subfield of $\mathbb{F}_{q^k}$, while $x_{P'}$ is not. Our results are significantly faster than other studies of this case, but nevertheless the cases with even embedding degree offer more advantages.

For curves of the form $y^2 = x^3 + b$, Lin *et al.* [LZZW08] observed that $1/v_{[2]R'}(P')$ can be written as

$$\frac{1}{v_{[2]R'}(P')} = \frac{1}{x_{P'} - x_{[2]R'}} = \frac{x_{P'}^2 + x_{P'}x_{[2]R'} + x_{[2]R'}^2}{(y_{P'} - y_{[2]R'})(y_{P'} + y_{[2]R'})}.$$

Since $(y_{P'} - y_{[2]R'})(y_{P'} + y_{[2]R'})$ lies in a subfield, the line function can be multiplied by $x_{P'}^2 + x_{P'}x_{[2]R'} + x_{[2]R'}^2$, instead of dividing it by $v_{[2]R'}(S)$. Analogously, the addition step becomes $\ell'_{R',Q'}(P') = \ell_{R',Q'}(P') \cdot (x_{P'}^2 + x_{P'}x_{Q'+R'} + x_{Q'+R'}^2)$.

3.4.1 Point doubling and line computation

In projective coordinates $x_{R'} = X_1/Z_1$ and $y_{R'} = Y_1/Z_1$, we replace $X_1^3 = Y_1^2 Z_1 - bZ_1^3$ and factor $\ell'_{[2]R'}(P')$ to see that $\ell'_{[2]R'}(P')$ equals

$$\alpha \cdot \left(X_1 Z_1 (Y_1^2 - 9bZ_1^2) \cdot x_{P'} + (4Y_1^2 Z_1^2) \cdot x_{P'}^2 - (6X_1^2 Y_1 Z_1) \cdot y_{P'} + X_1^2 (Y_1^2 + 9bZ_1^2) \right),$$

where $\alpha = (18bY_1^2 Z_1^2 - 27b^2 Z_1^4 + Y_1^4 + 8Y_1^3 Z_1 \cdot y_{P'})/(32Y_1^5 Z_1^3) \in \mathbb{F}_{q^{k/3}}$ does not contain $x_{P'}$ and can be discarded.

The values for X_1 and Z_1 are defined over subfields of $\mathbb{F}_{q^k}$ and we obtain more efficient formulas by computing $\ell''_{[2]R'}(P') = \ell'_{[2]R'}(P')X_1/(Z_1\alpha)$ as

$$\ell''_{[2]R'}(P') = X_1^2(Y_1^2 - 9bZ_1^2) \cdot x_{P'} + 4X_1 Y_1^2 Z_1 \cdot x_{P'}^2 - 6X_1^3 Y_1 \cdot y_{P'} + (Y_1^2 - bZ_1^2)(Y_1^2 + 9bZ_1^2).$$

For cubic twists, the term $x_{P'}^2 \in \mathbb{F}_{q^k}$ appears in the simplified doubling line function so we write $\ell''_{[2]R'}(P') = \ell_{1,0} \cdot x_{P'} + \ell_{2,0} \cdot x_{P'}^2 + \ell_{0,1} \cdot y_{P'} + \ell_{0,0}$. We compute $(X_3 : Y_3 : Z_3) = [2](X_1 : Y_1 : Z_1)$ and the necessary $\ell_{i,j}$ coefficients

using $6\mathbf{m} + 7\mathbf{s} + 1\mathbf{d}_b$ in addition to the multiplications by $x_{P'}, x_{P'}^2$, and $y_{P'}$.

$$\begin{aligned} A &= X_1^2,\ B = Y_1^2,\ C = Z_1^2,\ D = bC,\ E = 3D,\ F = (X_1 + Y_1)^2 - A - B, \\ G &= (Y_1 + Z_1)^2 - B - C,\ H = 3E,\ X_3 = F \cdot (B - H), \\ Y_3 &= (B + H)^2 - 3(2E)^2,\ Z_3 = 4B \cdot G,\ \ell_{1,0} = A \cdot (B - H),\ \ell_{2,0} = F \cdot G, \\ \ell_{0,1} &= -3A \cdot F,\ \ell_{0,0} = (B - D) \cdot (B + H). \end{aligned}$$

Note that the formulas in [MGI09] require $8\mathbf{m} + 9\mathbf{s} + 1\mathbf{d}_b$ in addition to the multiplications by $x_{P'}, x_{P'}^2$, $y_{P'}$, $y_{P'}^2$, $x_{P'}y_{P'}$, and $x_{P'}^2 y_{P'}$, i.e. they need 6 multiplications costing $k/3$ base field multiplications each while we only need 3 such multiplications. This means that the overall saving is $2\mathbf{m} + 2\mathbf{s} + k\mathbf{m}_1$.

3.4.2 Addition and line computation

For additions we break with the conventional wisdom that the line function should be given in terms of the base point. For even embedding degrees where denominator elimination does not require further adjustment, that approach is suitable and particularly helps if the base point is given in affine coordinates. For the curves in this section we show that building the line function on the resulting point $(X_3 : Y_3 : Z_3)$ gives better operation counts in spite of Z_3 not being equal to 1.

The default line function is given by $\left(\frac{(y_1-y_2)}{(x_1-x_2)} \cdot (x_1 - x_{P'}) + y_{P'} - y_1\right)/(x_3 - x_{P'})$. Using the above denominator elimination technique this gets transformed to

$$\left(\frac{(y_2 - y_1)}{(x_2 - x_1)} \cdot (x_1 - x_{P'}) + y_{P'} - y_1\right) \cdot \left(x_3^2 + x_3 x_{P'} + x_{P'}^2\right)/(y_3^2 - y_{P'}^2).$$

This approach leads to a polynomial of the form $\ell_{2,0} \cdot x_{P'}^2 + \ell_{1,0} \cdot x_{P'} + \ell_{1,1} \cdot x_{P'}y_{P'} + \ell_{2,1} \cdot x_{P'}^2 y_{P'} + \ell_{0,2} \cdot y_{P'} + \ell_{0,1} \cdot y_{P'} + \ell_{0,0}$ which requires $(6k/3)\mathbf{m}_1$ after the computation of the coefficients $\ell_{i,j}$.

In the representation

$$\left(\frac{(y_1 - y_2)}{(x_1 - x_2)} \cdot (x_3 - x_{P'}) + y_{P'} + y_3\right) \cdot \left(x_3^2 + x_3 x_{P'} + x_{P'}^2\right)/(y_3^2 - y_{P'}^2)$$

using the coordinates x_3, y_3 instead of x_1, y_1, it becomes obvious that the factor $(x_3 - x_{P'})(x_3^2 + x_3 x_{P'} + x_{P'}^2) = y_3^2 - y_{P'}^2$ appears in the left term of the numerator

and that thus the whole numerator is divisible by the subfield element $y_{P'} + y_3$. (Note the sign change on y_3 because the line goes through $(x_3, -y_3)$ by the geometric addition law on E.) This means that the line function is of the form $\ell_{2,0} \cdot x_{P'}^2 + \ell_{1,0} \cdot x_{P'} + \ell_{0,1} \cdot y_{P'} + \ell_{0,0}$, requiring only $(3k/3)\mathbf{m}_1$ after the computation of the coefficients $\ell_{i,j}$.

We obtain in projective coordinates that $\ell'_{R',Q'}(P')$ equals

$$\frac{(Y_1Z_2 - Y_2Z_1)Z_3(Y_3 - y_{P'}Z_3) + (X_3^2 + X_3Z_3x_{P'} + Z_3^2x_{P'}^2)(X_1Z_2 - X_2Z_1)}{(Y_3 - y_{P'}Z_3)(X_1Z_2 - X_2Z_1)Z_3}.$$

The denominator can be discarded. To compute the numerator more efficiently we observe that $Z_3 = Z_1Z_2(X_1Z_2 - X_2Z_1)^3$ so that we can divide by $(X_1Z_2 - X_2Z_1)$; furthermore we scale the function by 2 to allow an **s**-**m** tradeoff. This gives

$$\begin{aligned}\ell''_{R',Q'}(P') = {} & 2Z_3^2x_{P'}^2 + 2X_3Z_3x_{P'} - 2Z_1Z_2(X_1Z_2 - X_2Z_1)^2(Y_1Z_2 - Y_2Z_1)Z_3y_{P'} \\ & + 2X_3^2 + 2Z_1Z_2(X_1Z_2 - X_2Z_1)^2(Y_1Z_2 - Y_2Z_1)Y_3.\end{aligned}$$

We compute the addition and line computation using the following sequence of operations.

$$\begin{aligned}
A &= X_1 \cdot Z_2,\ B = Y_1 \cdot Z_2,\ C = Z_1 \cdot Z_2,\ D = Z_1 \cdot X_2 - A,\ E = B - Z_1 \cdot Y_2, \\
F &= D^2,\ G = E^2,\ H = -D \cdot F,\ I = F \cdot A,\ J = H + C \cdot G - 2I, \\
K &= C \cdot F \cdot E;\ X_3 = -D \cdot J,\ Y_3 = E \cdot (I - J) - (H \cdot B),\ Z_3 = C \cdot H, \\
L &= X_3^2,\ M = Z_3^2,\ N = (X_3 + Z_3)^2 - L - M,\ \ell_{2,0} = 2M,\ \ell_{1,0} = N, \\
\ell_{0,0} &= 2(L + K \cdot Y_3),\ \ell_{0,1} = -2K \cdot Z_3.
\end{aligned}$$

The explicit formulas for computing $(X_3 : Y_3 : Z_3)$ are the same as in the EFD [BL07a]; they use $12\mathbf{m} + 2\mathbf{s}$ and use the intermediate variables $A, \ldots, J$; the values $K, \ldots, N$ are used in the computation of the line function. The total operation count for the above sequence of operations is $16\mathbf{m} + 5\mathbf{s}$ in addition to the multiplications by $x_{P'}^2, x_{P'}$, and $y_{P'}$. Mixed addition is cheaper saving one $\mathbf{m}$ in each of A, B, and C and needing only $13\mathbf{m} + 5\mathbf{s}$.

In the pairing computation each addition is followed by a doubling. Thus $L = X_3^2$ and $M = Z_3^2$ should be cached and used in the doubling computation. This reuse reduces the effective costs of the addition step by $2\mathbf{s}$ and similarly for

the mixed-addition step. Accordingly we report $16\mathbf{m} + 3\mathbf{s}$ and $13\mathbf{m} + 3\mathbf{s}$ in the comparison in Section 3.6.

3.5 Fast pairings on special Weierstrass curves $y^2 = cx^3 + 1$

This section presents fast formulas that apply to curves of the form $y^2 = x^3 + c^2$, i.e. the curve constant must be a square in the field of definition of the curve. In this case we can define a trivial isomorphism σ between such a curve and the special Weierstrass curve $y'^2 = cx'^3 + 1$, defined by $\sigma : (x', y') \mapsto (cx', cy')$ and $\sigma^{-1} : (x, y) \mapsto (x/c, y/c)$. Thus, to apply these special formulas to the ate pairing, we would like the twisted curve E' to be of the form $y^2 = x^3 + c^2$ (see Section 3.1). Suppose we are employing a sextic twist of the curve $E/\mathbb{F}_q : y^2 = x^3 + b$, which (from Section 2.3.3) is of the form $E/\mathbb{F}_{q^{k/6}} : y^2 = x^3 + b\omega^6$, where ω is chosen so that (in particular) $\omega^3 \in \mathbb{F}_{q^{k/3}}$, but $\omega^3 \notin \mathbb{F}_{q^{k/6}}$. That is, $\omega^6 \in \mathbb{F}_{q^{k/6}}$ is chosen as a non-square. The only way our formulas can be applied is if $b\omega^6$ is a square in $\mathbb{F}_{q^{k/6}}$, from which it follows that our formulas apply if and only if b is non-square in $\mathbb{F}_{q^{k/6}}$. Since b is defined over the base field $\mathbb{F}_q$, it is clear that this can only happen if $12 \nmid k$, since otherwise b would automatically be a square over $\mathbb{F}_{q^{k/6}} = \mathbb{F}_{q^{2z}}$, for some $z \geq 1$. Thus rules out BN curves with $k = 12$ and BLS curves with $k = 24$. However, some attractive families that do have $6 \mid k$, $12 \nmid k$ and allow the curve constant to be non-square are the $k = 18$ KSS curves, and the $k = 6$ curves from [FST10, Const. 6.6]. In Section 3.5.3 we give an example from both families after we present the explicit formulas. Observe that when $12 \mid k$, then $c = \sqrt{b\omega^6} \notin \mathbb{F}_{q^{k/6}}$, so the isomorphism σ^{-1} will take a point (say Q') defined over the twisted subfield $\mathbb{F}_{q^{k/6}}$ into some larger extension field, which renders the powerful advantage of the sextic twist useless.

Assume then, that we are to compute the ate pairing $a_T(Q', P')$, where Q' and P' lie on the twist, which is the special Weierstrass curve $E' : y^2 = cx^3 + 1$, $c \in \mathbb{F}_{q^{k/d}}$. We work in homogeneous projective coordinates, so that the curve becomes $Y^2Z = cX^3 + Z^3$.

3.5.1 Point doubling and line computation

We start by observing that for curves of this special type the affine "schoolbook" doubling formulas in Eq. (2.5) can be rewritten conveniently. That is, to compute $(x_3, y_3) = [2](x_1, y_1)$ on $y^2 = cx^3 + 1$, we can take

$$\begin{aligned} x_3 &= x_1(\mu - \mu^2), \\ y_3 &= (y_1 - 1)\mu^3 - 1, \end{aligned}$$

where $\mu = (y_1 + 3)/(2y_1)$. Given $(X_1 : Y_1 : Z_1)$ with $Z_1 \neq 0$ the projective point doubling formulas for $[2](X_1 : Y_1 : Z_1) = (X_3 : Y_3 : Z_3)$ follow as

$$\begin{aligned} X_3 &= 2X_1Y_1(Y_1^2 - 9Z_1^2), \\ Y_3 &= (Y_1 - Z_1)(Y_1 + 3Z_1)^3 - 8Y_1^3Z_1, \\ Z_3 &= 8Y_1^3Z_1. \end{aligned}$$

The associated line formula for the doubling is

$$\ell_{R',R'}(P') = X_1^2(3cx_{P'}) - Y_1^2 + 3Z_1^2 - 2Y_1Z_1y_{P'}.$$

Since P' remains unchanged throughout the loop, precompute $\tilde{x}_{P'} = 3cx_{P'}$ and write $\ell_{R',R'} = \ell_{1,0}\tilde{x}_{P'} + \ell_{0,1}y_{P'} + \ell_{0,0}$. We compute the doubling and line computation using the following sequence of operations.

$$\begin{gathered} A = Y_1^2, \quad B = Z_1^2, \quad C = (Y_1 + Z_1)^2 - A - B, \quad \ell_{1,0} = X_1^2 \\ Z_3 = 4A \cdot C, \quad X_3 = 2X_1 \cdot Y_1 \cdot (A - 9B), \quad \ell_{0,1} = -2C, \\ Y_3 = (A - 3B + C) \cdot (A + 9B + 3C) - Z_3, \quad \ell_{0,0} = -A + 3B. \end{gathered}$$

The above formulas need a total of $3\mathbf{m} + 5\mathbf{s}$, which remains the fastest explicit formulas for the doubling step on any curve to date. Note that the $X_1 \cdot Y_1$ in the computation of X_3 can be computed with a squaring.

3.5.2 Point addition and line computation

In the case of addition, we found it most advantageous to simply amend the schoolbook formulas in Eq. (2.4). That is, to compute $(x_3, y_3) = (x_1, y_1) +$

(x_2, y_2), we take

$$\begin{aligned} x_3 &= c^{-1}\lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned}$$

where $\lambda = (y_1 - y_2)/(x_1 - x_2)$ as usual. Given $(X_1 : Y_1 : Z_1)$ and $(X_2 : Y_2 : Z_2)$ with $Z_1 \neq 0$ and $Z_2 \neq 0$ and $(X_1 : Y_1 : Z_1) \neq (X_2 : Y_2 : Z_2)$, an addition can be performed as $(X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2) = (X_3 : Y_3 : Z_3)$ where

$$\begin{aligned} X_3 &= (X_1Z_2 - Z_1X_2)(Z_1Z_2(Y_1Z_2 - Z_1Y_2)^2 - c(X_1Z_2 + Z_1X_2)(X_1Z_2 - Z_1X_2)^2), \\ Y_3 &= (Y_1Z_2 - Z_1Y_2)(c(2X_1Z_2 + Z_1X_2)(X_1Z_2 - Z_1X_2)^2 - Z_1Z_2(Y_1Z_2 - Z_1Y_2)^2) \\ &\quad -cY_1Z_2(X_1Z_2 - Z_1X_2)^3, \\ Z_3 &= cZ_1Z_2(X_1Z_2 - Z_1X_2)^3. \end{aligned}$$

The associated line formula is

$$\begin{aligned} \ell_{R,Q}(P') &= (Y_1Z_2 - Z_1Y_2)(X_2 - x_{P'}Z_2) - \\ &\quad (X_1Z_2 - Z_1X_2)Y_2 + (X_1Z_2 - Z_1X_2)Z_2y_{P'}. \end{aligned}$$

Since $Q = (X_2 : Y_2 : Z_2)$ is fixed, precompute $c_1 = X_2 - x_{P'}Z_2$. We compute the doubling and line computation using the following sequence of operations, where we also use t_1, t_2, t_3 and t_4 to provide register allocations.

$$\begin{gathered} t_1 = Z_1X_2, \quad X_3 = X_1Z_2, \quad t_1 = X_3 - t_1, \quad t_2 = Z_1Y_2, \quad Y_3 = Y_1Z_2, \\ t_2 = Y_3 - t_2, \quad \ell_{R,Q}(P') = c_1t_2 - t_1Y_2 + t_1Z_2y_{P'}, \quad Z_3 = Z_1Z_2, \\ t_3 = t_1^2, \quad t_3 = ct_3, \quad X_3 = t_3X_3, \quad t_3 = t_1t_3, \quad t_4 = t_2^2, \quad t_4 = t_4Z_3, \\ t_4 = t_3 + t_4, \quad t_4 = t_4 - X_3, \quad t_4 = t_4 - X_3, \quad X_3 = X_3 - t_4, \\ t_2 = t_2X_3, \quad Y_3 = t_3Y_3, \quad Y_3 = t_2 - Y_3, \quad X_3 = t_1t_4, \quad Z_3 = Z_3t_3, \end{gathered}$$

In the case of a full addition and line computation, the above formulas cost $13\mathbf{m} + 2\mathbf{s} + 1\mathbf{d}_c$. In the case of a mixed addition, i.e. where $Z_2 = 1$, the above formulas incur $10\mathbf{m} + 2\mathbf{s} + 1\mathbf{d}_c$. We mention a minor improvement that came later in our later work [CLN10] as follows. If the addition line in is written as $\ell'_{R',Q'}(P') = (Y_1Z_2 - Y_2Z_1) \cdot X_2 - (Y_1Z_2 - Y_2Z_1) \cdot x_{P'}Z_2 + (X_1Z_2 - X_2Z_1) \cdot y_{P'}Z_2 - (X_1Z_2 - X_2Z_1) \cdot Y_2$, the corresponding cost for an addition actually becomes $12\mathbf{m} + 2\mathbf{s}$. Note that the coefficients appear as subexpressions in the mixed addition of R' and Q', so computing $\ell'_{R',Q'}(P')$ as above costs an extra $(2k/d)\mathbf{m}_1 + 2\mathbf{m}$ for the Tate pairing and an extra $4\mathbf{m}$ for the ate pairing.

If $Q' = (X_2 : Y_2 : 1)$, the addition $R' + Q'$ becomes a mixed addition and costs $9\mathbf{m} + 2\mathbf{s}$. Computing the addition and the line as $\ell'_{m(R',Q')}(P') = (Y_1 - Y_2Z_1) \cdot X_2 - (Y_1 - Y_2Z_1) \cdot x_{P'} + (X_1 - X_2Z_1) \cdot y_{P'} - (X_1 - X_2Z_1) \cdot Y_2$ costs an extra $(2k/d)\mathbf{m}_1 + 2\mathbf{m}$ for both the Tate and the ate pairing. If Q' or P' is fixed in the mixed addition, similar comments to Section 3.2 apply, reducing the extra costs to only $(2k/d)\mathbf{m}_1 + \mathbf{m}$.

3.5.3 Example curves

We give two examples of families that facilitate an ate-like pairing with a sextic twist and make use of the explicit formulas for curves of the form $y^2 = x^3 + c^2$, which are slightly faster than the explicit formulas for the more general curves $y^2 = x^3 + b$.

Example 3.5.1. Construction 6.6 of [FST10] gives a family for $k = 6$ where the curve is of the form $y^2 = x^3 + b$, meaning that a sextic twist can be employed. The parameterisations are $q(x) = (x-1)^2(x^2 - x + 1)/3 + x$, $t(x) = x + 1$ and $r = x^2 - x + 1$, from which $x = 4$ gives a nice toy example with $q = 43$ and $r = 13$. The curve $E/\mathbb{F}_q : y^2 = x^3 + 2$ has $r \mid \#E = 52$, and we note that 2 is non-square in $\mathbb{F}_q$. We construct $\mathbb{F}_{q^k}$ as $\mathbb{F}_{q^6} = \mathbb{F}_q(u)$ where $u^6 - 3 = 0$. The correct sextic twist is given by $E'/\mathbb{F}_q : y^2 = x^3 + 6$ (the other option was $E'/\mathbb{F}_q : y^2 = x^3 + 2/3$), and since both 2 and 3 are non-square in $\mathbb{F}_q$, we can write the twist as $E'/\mathbb{F}_q : y^2 = x^3 + c^2$, where $c = \sqrt{6}$, so take $c = 7$. The curve $\tilde{E}'/\mathbb{F}_q : \tilde{y}^2 = c\tilde{x}^3 + 1$ is isomorphic to E' over $\mathbb{F}_q$ with $\sigma : \tilde{E}' \to E'$ defined as $\sigma : (\tilde{x}, \tilde{y}) \mapsto (c\tilde{x}, c\tilde{y})$. The point $Q' = (33, 9) \in E'$ is moved to $\tilde{Q}' \in \tilde{E}'$ via $\tilde{Q}' = \sigma^{-1}(33, 9) = (33/c, 9/c) = (33/7, 9/7) = (17, 32)$, both of which have order r on E' and $\tilde{E}'$ respectively.

Example 3.5.2. The KSS family with $k = 18$ can make use of the faster formulas on $y^2 = cx^3 + 1$. Indeed, the first such curve found (the parameterisations can be found in Chapter 6) for $x > 0$ is when $x = 3584$, which gives $q = 1298166528463937727281622301$ and $r = 6178938693718376449$, for which $E/\mathbb{F}_q : y^2 = x^3 + 2$ is such that $r \mid \#E$. We note that 2 is not a square in $\mathbb{F}_q$ and that this happens in general for this family. Thus, 2 will also not be square in $\mathbb{F}_{q^3}$, which we construct as $\mathbb{F}_{q^3} = \mathbb{F}_q(u)$, with $u^3 + 2 = 0$. Further, construct $\mathbb{F}_{q^{18}}$ as $\mathbb{F}_{q^{18}} = \mathbb{F}_{q^3}(v)$, with $v^6 - u = 0$. The correct twist is given as $E'/\mathbb{F}_{q^2} : y^2 = x^3 + 2/u$, and since both 2 and u are not square in $\mathbb{F}_{q^3}$, it follows that we can write the twist as $E'/\mathbb{F}_{q^3} : y^2 = x^3 + c^2$, where $c = \sqrt{2/u}$,

in this case giving $c = \pm 37928831032990325061756937 8u \in \mathbb{F}_{q^3}$. The curve $\tilde{E}'/\mathbb{F}_{q^3} : \tilde{y}^2 = c\tilde{x}^3 + 1$ is isomorphic to E' over $\mathbb{F}_{q^3}$ with $\sigma : \tilde{E}' \to E'$ defined as $\sigma : (\tilde{x}, \tilde{y}) \mapsto (c\tilde{x}, c\tilde{y})$.

3.6 Summary of contributions

This section compares the speed of our pairing formulas with the literature in the following categories:

- **(i)**: Curves of the form $y^2 = x^3 + ax$ have twists of degree $d = 2$ and 4. We compare our operation counts from Section 3.2 with the results given by Ionica and Joux [IJ08] and Arène *et al.* [ALNR10]; note that those papers cover general Weierstrass curves but we are not aware of any other study covering this case.

- **(ii)**: Curves of the form $y^2 = x^3 + c^2$ have a point of order 3 and can admit twists of degrees $d = 2$ and 6. Our formulas in Section 3.5 are applicable for the case of sextic twists when $6 \mid k$ but $12 \nmid k$, i.e. $k = 6, 18, 30$, etc. These are the fastest formulas for pairings across any curve model, improving slightly on our formulas for the more general case in (iii).

- **(iii)**: Curves of the form $y^2 = x^3 + b$ do not necessarily have a point of order 3. We study operation counts for twists of degree 2 and 6. These curves cover in particular BN curves [BN05]. We compare our new formulas from Section 3.3 with those given for the same curve shape in [ALNR10].

- **(iv)** Curves of the form $y^2 = x^3 + b$ also have twists of degree 3. This case requires very different optimisations and has not been studied much in the literature. The first paper studying pairing computation on curves admitting cubic twists [LZZW08] did not pay close attention to the operation count itself, so we compare our formulas from Section 3.4 with the results presented by El Mrabet *et al.* in [MGI09], although that paper did not present addition formulas.

The above papers for even d give $k\mathbf{m}_1$ for evaluating the line function. This can almost always be done in $(2k/d)\mathbf{m}_1$, so we adjust their results accordingly. In the general addition case, Table 3.2 only gives counts for the Tate pairing. For $d = 2$ it is possible to save $1\mathbf{m}$ in each ADD and each mADD. For the ate

pairing in this case the costs are different and the operation counts should be modified by $-(2k/d)\mathbf{m}_1 + 2\mathbf{m}$.

For mixed additions we use our improved precomputations, assuming that one of the input points is fixed.

Curve Curve order Twist deg.	Best Coord.	DBL ADD mADD	Prev. best Coord.	DBL ADD mADD
$y^2 = x^3 + ax$ - $d = 2, 4$	Sec. 3.2 $\mathcal{W}_{(1,2)}$	$(2k/d)\mathbf{m}_1 + 2\mathbf{m} + 8\mathbf{s} + 1\mathbf{d}_a$ $(2k/d)\mathbf{m}_1 + 12\mathbf{m} + 7\mathbf{s}$ $(2k/d)\mathbf{m}_1 + 9\mathbf{m} + 5\mathbf{s}$	[IJ08], [ALNR10] $\mathcal{J}$	$(2k/d)\mathbf{m}_1 + 1\mathbf{m} + 11\mathbf{s} + 1\mathbf{d}_a$ $(2k/d)\mathbf{m}_1 + 10\mathbf{m} + 6\mathbf{s}$ $(2k/d)\mathbf{m}_1 + 7\mathbf{m} + 6\mathbf{s}$
$y^2 = x^3 + c^2$ $3 \mid \#E$ $d = 2, 6$	Sec. 3.5 $\mathcal{P}$	$(2k/d)\mathbf{m}_1 + 3\mathbf{m} + 5\mathbf{s}$ $(2k/d)\mathbf{m}_1 + 14\mathbf{m} + 2\mathbf{s} + 1\mathbf{d}_c$ $(2k/d)\mathbf{m}_1 + 10\mathbf{m} + 2\mathbf{s} + 1\mathbf{d}_c$	[ALNR10] $\mathcal{P}$	$(2k/d)\mathbf{m}_1 + 3\mathbf{m} + 8\mathbf{s}$ $(2k/d)\mathbf{m}_1 + 10\mathbf{m} + 6\mathbf{s}$ $(2k/d)\mathbf{m}_1 + 7\mathbf{m} + 6\mathbf{s}$
$y^2 = x^3 + b$ $3 \nmid \#E$ $d = 2, 6$	Sec. 3.3 $\mathcal{P}$	$(2k/d)\mathbf{m}_1 + 2\mathbf{m} + 7\mathbf{s} + 1\mathbf{d}_b$ $(2k/d)\mathbf{m}_1 + 14\mathbf{m} + 2\mathbf{s}$ $(2k/d)\mathbf{m}_1 + 10\mathbf{m} + 2\mathbf{s}$	[ALNR10] $\mathcal{J}$	$(2k/d)\mathbf{m}_1 + 3\mathbf{m} + 8\mathbf{s}$ $(2k/d)\mathbf{m}_1 + 10\mathbf{m} + 6\mathbf{s}$ $(2k/d)\mathbf{m}_1 + 7\mathbf{m} + 6\mathbf{s}$
$y^2 = x^3 + b$ - $d = 3$	Sec. 3.4 $\mathcal{P}$	$k\mathbf{m}_1 + 6\mathbf{m} + 7\mathbf{s} + 1\mathbf{d}_b$ $k\mathbf{m}_1 + 16\mathbf{m} + 3\mathbf{s}$ $k\mathbf{m}_1 + 13\mathbf{m} + 3\mathbf{s}$	[MGI09] $\mathcal{P}$	$2k\mathbf{m}_1 + 8\mathbf{m} + 9\mathbf{s} + 1\mathbf{d}_b$ *ADD/mADD* *not reported*

Table 3.2: Comparisons of our pairing formulas with the previous fastest formulas.

We point out that all new doublings are faster than the previous ones. In (i), (ii) and (iii) this comes at the expense of somewhat slower additions. As we saw in Section 2.6.5 however, doublings are significantly more frequent than additions so that this disadvantage is amply mitigated by the faster doublings. Since many other non-Weierstrass curve models (e.g. Edwards, Hessian, Jacobi-Quartics, etc) achieved great success in ECC, formulas for pairing computation on these curves have also been developed [IJ08, DS08, ALNR10, JTV10]. Whilst such curve models have superseded Weierstrass curves in the context of elliptic curve operations only, the Weierstrass formulas presented in this chapter remain the fastest pairing formulas in all cases of practical relevance. The simple chord-and-tangent description of the group law means that the pairing functions are inherent in the elliptic curve operations, and extracting them in the Miller loop incurs minimal overhead.

Finally, we mention that Aranha *et al.* [AKL+11] used a slightly modified version of our formulas in their record-breaking pairing implementation. In deriving our formulas, we greatly prioritised squeezing as many field multiplications out of the routine as possible, achieving 2 multiplications and 7 squarings in case (iii)

above. They reversed one of the $\mathbf{m} - \mathbf{s}$ trade-offs that our formulas used, and in doing so, computed with 3 multiplications and 6 squarings, but with 6 less field additions. Since their implementation had that $\mathbf{m} - \mathbf{s} \approx 3\mathbf{a}$ (the difference between a multiplication and a squaring being roughly 3 additions), this reversal was a favourable trade-off. We note that most of our formulas offer room for these sorts of tweaks, and that serious implementors should have a good gauge on their field operation complexities before looking for such modifications.

Chapter 4

Loop Unrolling in Miller's Algorithm

This chapter shows that significant speedups can be achieved in pairing computations by unrolling consecutive iterations in the Miller loop. We show that our method gives faster Tate pairings for large embedding degrees, whilst for one-off ate pairings it is preferable not to unroll. When we get to Section 4.4 however, we show that when pairings have a fixed argument that facilitates precomputations, our method of unrolling becomes especially preferable for ate-like pairings. Thus, as far as improving the state-of-the-art, up to and including Section 4.3 of this chapter (which focusses on Tate pairings) can be seen as preparation for the fixed argument scenario in Section 4.4.

For now, assume that we are computing a Tate-like pairing as $e(P,Q) = f_{m,P}(Q)^{(q^k-1)/r}$. Notice that we write the loop length here as m, which is not necessarily the subgroup order r, since the *twisted ate pairing* of [HSV06] (which we did not discuss in Chapter 2) is a loop shortened variant of the Tate pairing on $\mathbb{G}_1 \times \mathbb{G}_2$. In most cases the twisted ate pairing can achieve $\log_2 m = \log_2 r/2$, so it is computed like a Tate pairing with half the loop length. The important thing to carry with us until Section 4.4 is that the first argument in the pairing is now $P \in \mathbb{G}_1$, which is defined over the base field $\mathbb{F}_q$. The second argument Q is defined over the full extension field. This is the setting we were in throughout Chapter 2 prior to reversing the roles of P and Q for the ate pairing description, so refer back to Algorithm 2.2 for the BKLS-GHS version of Miller's algorithm

that applies to this particular scenario.

The idea of this work is to exploit the fact that although the function updates ℓ in Algorithm 2.2 evaluate to give an element in $\ell(Q) \in \mathbb{F}_{q^k}$, they are in $\mathbb{F}_q[x, y]$ before they are evaluated, i.e. the (indeterminate) Miller functions $\ell(x, y)$ have coefficients that come from the ground field $\mathbb{F}_q$. Miller's algorithm absorbs the evaluation of these functions at Q into the paired value f before performing further operations in $\mathbb{F}_{q^k}$, such as $f \leftarrow f^2$. However, we propose that it can be advantageous to operate on $\ell(x, y)$ *before* evaluating it at the point Q. In this way we will be performing a lot more arithmetic in $\mathbb{F}_q$, but at the same time we avoid computations in $\mathbb{F}_{q^k}$. We argue that if k is big enough, so that the complexity of operations $\mathbb{F}_{q^k}$ is far greater than that of analogous operations in $\mathbb{F}_q$, then this trade-off can significantly favour our cause to obtain a faster Tate pairing.

Eisenträger, Lauter and Montgomery [ELM03] managed to avoid some full extension field arithmetic in pairing computations by combining two linear Miller functions into a single function of degree 2, which they call a parabola, and achieving a speedup by replacing two multiplications by the two linear functions with a single multiplication by the parabola. However, their algorithm has limited application in modern pairing implementations because it only applies to the point addition stage of Miller's algorithm, and (as we saw in Section 2.6.5) modern implementations will use low Hamming weight loop parameters that minimise the occurrence of these additions. Blake, Murty and Xu [BKMX06] extended the work in [ELM03] to form combinations of Miller lines that apply to every iteration of the Miller loop. However, their techniques are again not optimised for modern implementations because the algorithm proposed in [BKMX06] targeted removing field divisions, a problem that has since became obsolete thanks to the denominator elimination technique (see Section 2.6.1).

Our work in this chapter extends the notion of combining Miller lines into higher degree polynomials. In the next section we present our general approach, called Miller 2^n-tuple-and-add, which is analogous to the 2^n-ary windowing methods for general exponentiation (cf. [ACD+05, §9]), for an arbitrary window of size n. Our idea mimics that of Granger, Page and Stam [GPS06, §6], who employ *loop unrolling* to combine two Miller iterations at a time in the case of characteristic three pairing implementations. We apply this same technique to the wider and more common case of pairings over large prime fields, and consider unrolling

n consecutive iterations into one, for which we find that $n \leq 2$ is essentially optimal for large embedding degrees. For cases where $n = 2$ is better than $n = 1$, this results in a non-trivial speed-up in pairing implementations.

We organise this chapter as follows. We present the general algorithm description in Section 4.1. In Section 4.2 we focus on obtaining optimised explicit formulas for the special case of $n = 2$, which we call the *quadruple-and-add* version of Miller's algorithm; we demonstrate this algorithm in Section 4.2.3 where we give a detailed example. In Section 4.3 we give an overview of the case when $n = 3$, which we call the *octuple-and-add* version of Miller's algorithm, and for which the full explicit formulas can be found in [CBNW10a, A.3,A.4]. Although we do not recommend these octupling formulas for one-off Tate or ate pairings, their practical application becomes relevant in Section 4.4 when we consider the fixed argument scenario.

4.1 Miller 2^n-tuple-and-add

In this section we give a general unrolled version of Miller's algorithm by combining n consecutive doubling steps into one 2^n-tupling step. For any n, we naturally refer to the following process as the Miller 2^n-tuple-and-add algorithm. Suppose that at some stage of Algorithm 2.2, we have the intermediate multiple of P being $R = [v]P$, and the intermediate Miller function being $f_{v,P}$ (ignore the evaluation at Q for now). Consider the next n consecutive squarings on the function $f_{v,P}$, which equates to raising $f_{v,P}$ to the power 2^n. The divisor of the resulting function is

$$\begin{aligned} \mathrm{div}\big((f_{v,P})^{2^n}\big) &= 2^n \cdot \mathrm{div}(f_{v,P}), \\ &= 2^n v(P) - 2^n([v]P) - 2^n(v-1)(\mathcal{O}). \end{aligned} \tag{4.1}$$

To obtain the desired Miller function $f_{2^n v,P}$ from $f_{v,P}$, we must now find a function ℓ^* such that $\mathrm{div}((f_{v,P})^{2^n}) + \mathrm{div}(\ell^*) = \mathrm{div}(f_{2^n v,P}) = 2^n v(P) - ([2^n v]P) - (2^n v - 1)(\mathcal{O})$. We construct ℓ^* as

$$\ell^* = \prod_{i=1}^{n} (\ell_{[2^{i-1}]R,[2^{i-1}]R})^{2^{n-i}}, \tag{4.2}$$

the divisor of which is

$$\begin{aligned} \mathrm{div}(\ell^*) &= \sum_{i=1}^{n} 2^{n-i} \cdot \mathrm{div}(\ell_{[2^{i-1}]R,[2^{i-1}]R}), \\ &= \sum_{i=1}^{n} 2^{n-i} \cdot (2([2^{i-1}]R) - ([2^i]R) - (\mathcal{O})), \\ &= 2^n(R) - ([2^n]R) - (2^n - 1)(\mathcal{O}). \end{aligned} \tag{4.3}$$

Substituting $R = [v]P$ into (4.3) and combining this with (4.1) reveals that $\mathrm{div}((f_{v,P})^{2^n}) + \mathrm{div}(\ell^*) = \mathrm{div}(f_{2^n v,P})$, so that ℓ^* is indeed the required function. We note that the construction of ℓ^* is intuitive. Namely, ℓ^* is simply the product of the n different g's that are formed throughout each of the n equivalent double-and-add iterations, each of which accumulates a different exponent depending on how many squarings it encounters in the iterations that follow. In this light, Miller 2^n-tuple-and-add is much the same as Miller double-and-add; the above derivation (for general n) is analogous to the derivation (for $n = 1$) we saw back in Section 2.4.3. The major difference is that in Miller 2^n-tuple-and-add we do not multiply the Miller function by its update g immediately after it is squared. Rather, we form a product ℓ^* of n powers of such g's and we delay the multiplication of ℓ^* by f so that it occurs only once in what is the equivalent of n double-and-add iterations. This is simply Miller's algorithm with a window of size n.

For the addition step in the Miller 2^n-tuple-and-add algorithm, we now have to consider adding some multiple $[w]P$ of P ($w < 2^n$) to the intermediate point and updating the Miller function accordingly. Suppose that the intermediate point is $R = [v]P$ and the related Miller function prior to the addition has divisor $\mathrm{div}(f_{v,P}) = v(P) - ([v]P) - (v-1)(\mathcal{O})$ as before. We require a function ℓ^+ such that $\mathrm{div}(f_{v,P}) + \mathrm{div}(\ell^+) = \mathrm{div}(f_{(v+w),P}) = (v+w)(P) - ([v+w]P) - (v+w-1)(\mathcal{O})$. The straightforward way to construct such a function is

$$\ell^+ = \prod_{i=0}^{w-1} \ell_{R+[i]P,P}, \tag{4.4}$$

the divisor of which is

$$\begin{aligned}\operatorname{div}(\ell^+) &= \sum_{i=0}^{w-1} \operatorname{div}(\ell_{R+[i]P,P}) \\ &= \sum_{i=0}^{w-1} \big[(P) + (R+[i]P) - (R+[i+1]P) - (\mathcal{O})\big], \\ &= w(P) + (R) - (R+[w]P) - w(\mathcal{O}).\end{aligned}$$

Again, substituting $R = [v]P$ gives $\operatorname{div}(\ell^+) = w(P) + ([v]P) - ([v+w]P) - w(\mathcal{O})$, so that $\operatorname{div}(f_{v,P}) + \operatorname{div}(\ell^+) = \operatorname{div}(f_{(v+w),P})$, and we see that ℓ^+ is clearly the desired function. However, if we compute ℓ^+ in the above fashion, we have to compute the product of w different addition lines, and since w can take any value between 1 and $2^n - 1$, computing the addition step with the explicit formulas that result from the product in (4.4) can become quite costly. Instead, consider an alternative method of computing the addition update function as follows. Let ℓ^+_{alt} be such that $\operatorname{div}(\ell^+_{\text{alt}}) = \operatorname{div}(\ell^+)$ and take

$$\ell^+_{\text{alt}} = f_{w,P} \cdot \ell_{[v]P,[w]P}, \tag{4.5}$$

so that $\operatorname{div}(\ell^+_{\text{alt}}) = \operatorname{div}(f_{w,P}) + \operatorname{div}(\ell_{[v]P,[w]P}) = w(P) + ([v]P) - ([v+w]P) - w(\mathcal{O})$. The advantage of the computation of ℓ^+_{alt} over the computation of ℓ^+ is that ℓ^+_{alt} is comprised of only two functions, regardless of the size of w. Moreover, the function $f_{w,P}$ is the same function throughout the entire Miller 2^n-tupling loop and does not change depending on where the addition/s occurs. Thus, the $f_{w,P}$'s can be precomputed (for all necessary values of w) prior to entering the Miller 2^n-tupling loop so that we must only construct one new line function ($\ell_{[v]P,[w]P}$) at each addition stage. Importantly, this addition line is computed by applying the standard addition formulas to the coordinates of the point $[v]P$, which changes in each iteration, and the point $[w]P$ whose coordinates can be cached initially. From here on, the construction of ℓ^+ refers to the construction of ℓ^+_{alt} described in (4.5). We summarise in Algorithm 4.1, where we note that the first value in the base 2^n representation of m will not be $m_{l-1} = 1$ in general, so that we begin with an addition before entering the loop when $m_{l-1} \neq 1$.

Algorithm 4.1 Miller 2^n-tuple-and-add Algorithm.

Input: P, Q, $m = (m_{l-1}...m_1, m_0)_{2^n}$, and the necessary precomputed values of $w[P]$ where $w < 2^n$.
Output: $f_{m,P}(Q)$.

1: Compute and evaluate function $f_{w,P}(Q)$ with $w = m_{l-1}$.
2: $f \leftarrow f_{w,P}(Q)$.
3: $R \leftarrow [m_{l-1}]P$.
4: **for** $i = l - 2$ to 0 **do**
5: Compute function ℓ^* in the 2^n-tupling of R.
6: $R \leftarrow [2^n]R$.
7: $f \leftarrow f^{2^n} \cdot \ell^*$.
8: **if** $m_i \neq 0$ **then**
9: Compute function ℓ^+ as the product described in (4.5) with $w = m_i$.
10: $R \leftarrow R + [m_i]P$.
11: $f \leftarrow f \cdot \ell^+$.
12: **end if**
13: **end for**
14: **return** f.

4.2 Quadruple-and-add

In this section we focus on applying the algorithm from the previous section in the specific case that $n = 2$. We present reduced explicit formulas that arise for the Miller quadruple-and-add algorithm on curves of the form $E : y^2 = x^3 + b$ and $E : y^2 = x^3 + ax$, since these are the most efficient curve shapes used in practice (refer to Section 2.5). We focus solely on the quadrupling stage of the algorithm (i.e. steps 6 and 7 in Algorithm 4.1), since optimised loop parameters will result in very few additions (refer back to Section 2.6.5).

We begin by setting $n = 2$ in (4.2) to obtain the Miller update ℓ^* corresponding to the quadrupling of R as

$$\ell^* = \prod_{i=1}^{2} (\ell_{[2^{i-1}]R,[2^{i-1}]R})^{2^{2-i}} = \left(\ell_{R,R}\right)^2 \cdot \left(\ell_{[2]R,[2]R}\right),$$

which has divisor $4(R) - ([4]R) - 3(\mathcal{O})$.

The recipe for obtaining explicit formulas for the function ℓ is straightforward. The functions $\ell_{R,R} = \lambda x + \nu$ and $\ell_{[2]R,[2]R} = \lambda' x + \nu'$ have $\lambda, \lambda', \nu, \nu'$ entirely dependent on the coordinates of $R = (x_1, y_1)$, so we can obtain inversion-free formulas via the substitution corresponding to a particular projection. We then look for factors that are contained in subfields (i.e. factors that do not contain y,

since $y_Q \in \mathbb{F}_{q^k}$), and that can therefore be removed (refer to Section 2.6.1). At any stage we can choose to apply variants of Gröbner basis reduction (modulo the elliptic curve equation) to simplify formulas as much as possible. Finally, once we are satisfied with the simplification of these formulas, we can begin determining a fast route to computing them.

4.2.1 Quadruple-and-add on $y^2 = x^3 + b$

For curves of this form, the fastest explicit formulas for the $n = 1$ case were derived using homogeneous projective coordinates in Section 3.3. We found that these coordinates also give the best results for $n \geq 2$, so substituting $x_1 = X_1/Z_1$ and $y_1 = Y_1/Z_1$ into the affine version of ℓ^* gives

$$\ell^* = \alpha \cdot (\ell_{1,0} \cdot x_Q + \ell_{2,0} \cdot x_Q^2 + \ell_{0,1} \cdot y_Q + \ell_{1,1} \cdot x_Q y_Q + \ell_{0,0}),$$

where $\alpha = -Z_1^3(X_1(X_1^3 - 8bZ_1^3) - 4Z_1(X_1^3 + bZ_1^3) \cdot x_Q)^2/(64Z_1^7Y_1^5(27X_1^6 - 36X_1^3Y_1^2Z_1 + 8Y_1^4Z_1^2))$ can be eliminated to give $\hat{\ell}^* = \ell^*/\alpha$, and where the $\ell_{i,j}$ coefficients are

$\ell_{2,0} = -6X_1^2Z_1(5Y_1^4 + 54bY_1^2Z_1^2 - 27b^2Z_1^4)$; $\quad \ell_{0,1} = 8X_1Y_1Z_1(5Y_1^4 + 27b^2Z_1^4)$;
$\ell_{1,1} = 8Y_1Z_1^2(Y_1^4 + 18bY_1^2Z_1^2 - 27b^2Z_1^4)$; $\qquad \ell_{0,0} = 2X_1(Y_1^6 - 75bY_1^4Z_1^2 + 27b^2Y_1^2Z_1^4 - 81b^3Z_1^6)$;
$\ell_{1,0} = -4Z_1(5Y_1^6 - 75bZ_1^2Y_1^4 + 135Y_1^2b^2Z_1^4 - 81b^3Z_1^6)$.

We let $(X_{D^2} : Y_{D^2} : Z_{D^2}) = [2](X_{D^1} : Y_{D^1} : Z_{D^1}) = [4](X_1 : Y_1 : Z_1)$, and compute the first doubling with small extra computation as

$$X_{D^1} = 4X_1Y_1(Y_1^2 - 9bZ_1^2), \;\; Y_{D^1} = 2Y_1^4 + 36bY_1^2Z_1^2 - 54b^2Z_1^4, \;\; Z_{D^1} = 16Y_1^3Z_1.$$

The calculation of the $\ell_{i,j}$ coefficients and the intermediate point $(X_{D^1} : Y_{D^1} : Z_{D^1}) = [2](X_1, Y_1, Z_1)$ requires $11\mathbf{m}_1 + 11\mathbf{s}_1 + 3\mathbf{d}$ – see the sequence of operations below. To calculate $(X_{D^2} : Y_{D^2} : Z_{D^2}) = [4](X_1, Y_1, Z_1)$, we double the point $(X_{D^1} : Y_{D^1} : Z_{D^1})$ using the doubling formulas from Section 3.3 which cost $3\mathbf{m}_1 + 5\mathbf{s}_1 + 1\mathbf{d}$. The multiplication of each of the four $\ell_{i,j} \neq \ell_{0,0}$ by $x_Q^i y_Q^j$ costs $e\mathbf{m}_1$ (see the previous Chapter). Thus, the total cost for the quadrupling stage (not including the function update) is $14\mathbf{m}_1 + 16\mathbf{s}_1 + 4e\mathbf{m}_1 + 4\mathbf{d}$, plus many $\mathbb{F}_q$ additions that occur in the sequence below.

$A = Y_1^2$, $B = Z_1^2$, $C = A^2$, $D = B^2$, $E = (Y_1 + Z_1)^2 - A - B$, $F = E^2$, $G = X_1^2$, $H = (X_1 + Y_1)^2 - A - G$, $I = (X_1 + E)^2 - F - G$, $J = (A + E)^2 - C - F$, $K = (Y_1 + B)^2 - A - D$, $L = 27b^2D$, $M = 9bF$, $N = A \cdot C$, $R = A \cdot L$, $S = bB$, $T = S \cdot L$, $U = S \cdot C$, $X_{D^1} = 2H \cdot (A - 9S)$, $Y_{D^1} = 2C + M - 2L$, $Z_{D^1} = 4J$, $\ell_{1,0} = -4Z_1 \cdot (5N + 5R - 3T - 75U)$, $\ell_{2,0} = -3G \cdot Z_1 \cdot (10C + 3M - 2L)$, $\ell_{0,1} = 2I \cdot (5C + L)$, $\ell_{1,1} = 2K \cdot Y_{D^1}$, $\ell_{0,0} = 2X_1 \cdot (N + R - 3T - 75U)$. $F^* = \ell_{1,0} \cdot x_P + \ell_{2,0} \cdot x_P^2 + \ell_{0,1} \cdot y_P + \ell_{1,1} \cdot x_P y_P + \ell_{0,0}$, $A_2 = Y_{D^1}^2$, $B_2 = Z_{D^1}^2$, $C_2 = 3bB_2$, $D_2 = 2X_{D^1} \cdot Y_{D^1}$, $E_2 = (Y_{D^1} + Z_{D^1})^2 - A_2 - B_2$, $F_2 = 3C_2$, $X_{D^2} = D_2 \cdot (A_2 - F_2)$, $Y_{D^2} = (A_2 + F_2)^2 - 12C_2^2$, $Z_{D^2} = 4A_2 \cdot E_2$.

A magma script of the above routine is in [CBNW10a, §B].

4.2.2 Quadruple-and-add on $y^2 = x^3 + ax$

For curves of this shape, the fastest formulas for the standard double-and-add case were derived in weight-$(1, 2)$ coordinates in Section 3.2. Again, our experiments agree with these coordinates for $n \geq 2$, so we substitute $x_1 = X_1/Z_1$ and $y_1 = Y_1/Z_1^2$ into the affine version of ℓ^* to give

$$\ell^* = \alpha \cdot (\ell_{1,0} \cdot x_Q + \ell_{2,0} \cdot x_Q^2 + \ell_{0,1} \cdot y_Q + \ell_{1,1} \cdot x_Q y_Q + \ell_{0,0}),$$

where $\alpha = -Z_1^6(-4X_1Z_1(X_1^2 + aZ_1^2)x_Q + (X_1^2 - aZ_1^2)^2)^2$ can be eliminated to give $\hat{\ell}^* = \ell^*/\alpha$, and where the $\ell_{i,j}$ coefficients are

$$\begin{aligned}
\ell_{1,0} &= -2X_1Z_1(5X_1^8 + 4aX_1^6Z_1^2 + 38a^2X_1^4Z_1^4 + 20a^3X_1^2Z_1^6 - 3a^4Z_1^8);\\
\ell_{2,0} &= -Z_1^2(15X_1^8 + 68aX_1^6Z_1^2 + 10a^2X_1^4Z_1^4 - 28a^3X_1^2Z_1^6 - a^4Z_1^8);\\
\ell_{0,1} &= 4Y_1X_1Z_1(5X_1^6 + 13aX_1^4Z_1^2 + 15a^2X_1^2Z_1^4 - a^3Z_1^6);\\
\ell_{1,1} &= 4Y_1Z_1^2(X_1^2 - aZ_1^2)(X_1^4 + 6aX_1^2Z_1^2 + a^2Z_1^4);\\
\ell_{0,0} &= X_1^2(X_1^8 - 20aX_1^6Z_1^2 - 26a^2X_1^4Z_1^4 - 20a^3X_1^2Z_1^6 + a^4Z_1^8).
\end{aligned}$$

Again, we compute the first doubling with small extra computation as

$$X_{D^1} = (X_1^2 - aZ_1^2)^2,\ Y_{D^1} = 2Y_1(X_1^2 - aZ_1^2)(X_1^4 + 6X_1^2aZ_1^2 + a^2Z_1^4),\ Z_{D^1} = 4Y_1^2.$$

The calculation of the $\ell_{i,j}$ coefficients and the intermediate point $(X_{D^1} : Y_{D^1} : Z_{D^1}) = [2](X_1, Y_1, Z_1)$ requires $10\mathbf{m} + 14\mathbf{s} + 2\mathbf{d}$ – see the sequence of operations below. To calculate $(X_{D^2} : Y_{D^2} : Z_{D^2}) = [4](X_1, Y_1, Z_1)$, we double the point $(X_{D^1} : Y_{D^1} : Z_{D^1})$ using the doubling formulas in Section 3.3 which cost $1\mathbf{m} + 6\mathbf{s} + 1\mathbf{d}$. Thus, the total cost for the quadrupling stage (not including the function update) is $11\mathbf{m}_1 + 20\mathbf{s}_1 + 4e\mathbf{m}_1 + 3\mathbf{d}$, as well as many $\mathbb{F}_q$ additions that are required

for computing the sequence below.

$$A = X_1^2, \quad B = Y_1^2, \quad C = Z_1^2, \quad D = aC, \quad X_{D1} = (A - D)^2, \quad E = 2(A + D)^2 - X_{D1}, \quad F = ((A - D + Y_1)^2 - B - X_{D1}),$$
$$Y_{D1} = E \cdot F, \quad Z_{D1} = 4B, \quad G = A^2, \quad H = D^2, \quad I = G^2, \quad J = H^2, \quad K = (X_1 + Z_1)^2 - A - C, \quad L = K^2,$$
$$M = (Y_1 + K)^2 - L - B, \quad N = ((G + H)^2 - I - J), \quad R = aL, \quad S = R \cdot G, \quad T = R \cdot H, \quad \ell_{1,1} = 2C \cdot Y_{D1},$$
$$\ell_{0,1} = M \cdot (5A \cdot (G + 3H) + D \cdot (13G - H)), \quad \ell_{2,0} = -C \cdot (15I + 17S + 5N - 7T - J),$$
$$\ell_{1,0} = -K \cdot (5I + S + 19N + 5T - 3J), \ell_{0,0} = A \cdot (I - 5S - 13N - 5T + J), \quad B_2 = Y_{D1}^2$$
$$F^* = \ell_{1,0} \cdot x_P + \ell_{2,0} \cdot x_P^2 + \ell_{0,1} \cdot y_P + \ell_{1,1} \cdot x_P y_P + \ell_{0,0}, A_2 = X_{D1}^2, \quad C_2 = Z_{D1}^2, \quad D_2 = aC_2, \quad X_{D2} = (A_2 - D_2)^2,$$
$$E_2 = 2(A_2 + D_2)^2 - X_{D2}, \quad Z_{D2} = 4B_2, \quad F_2 = ((A_2 - D_2 + Y_{D1})^2 - B_2 - X_{D2}, \quad Y_{D2} = E_2 \cdot F_2.$$

4.2.3 A detailed example

Since our original paper did not give a working example, we give one here. We choose a $k = 24$ BLS curve for illustrative purposes, since this gives a favourable ratio when trading $\mathbb{F}_{q^k}$ multiplications for multiplications in $\mathbb{F}_q$. Turn back to Example 2.6.9 or Example 2.6.11 to see the parameterisations for this family, for which the smallest BLS curve is found with $x = -5$, giving $q = 4680007$ and $r = 390001$. We can form the extension field using the tower

$$\mathbb{F}_q \xrightarrow{u^2+1} \mathbb{F}_{q^2} \xrightarrow{v^2-(u+3)} \mathbb{F}_{q^4} \xrightarrow{w^3-v} \mathbb{F}_{q^{12}} \xrightarrow{z^2-w} \mathbb{F}_{q^{24}}.$$

$E/\mathbb{F}_q : y^2 = x^3 + 1$ is easily seen to be the curve with $r \mid \#E(\mathbb{F}_q)$, and the correct sextic twist is $E'/\mathbb{F}_{q^4} : y^2 = x^3 + v$. We can generate $P \in \mathbb{G}_1$ by taking a random point in $E(\mathbb{F}_q)$ and multiplying by the cofactor $h = \#E/r = 12$. To generate $Q \in \mathbb{G}_2$, we take a random point in $E'(\mathbb{F}_{q^4})$, multiply by the cofactor $h' = 1230042989266471802425$ to get $Q' \in \mathbb{G}_2'$, and untwist via the isomorphism Ψ to get $Q = \Psi(Q')$. With our extension field tower, Ψ is defined as $\Psi : E' \to E$, where $\Psi : (x', y') \mapsto (x'/w, y'/(wz))$. Thus, let P and $Q = (x_Q, y_Q)$ be given as

$$\begin{aligned} P &= (577155, 707379); \\ x_Q &= \big((1831231u + 994504)v + (4652492u + 671306)\big)w^2; \\ y_Q &= \big((3495438u + 1355667)v + (763329u + 1210930)\big)wz. \end{aligned}$$

We will compute the Tate pairing $e(P, Q) = f_{r,P}(Q)^{(q^k-1)/r}$ using both the standard (double-and-add) and the quadruple-and-add versions of Miller's algorithm, i.e. using Algorithm 2.2 and Algorithm 4.1 respectively. To do so, we write both

the binary and quarternary representations of the loop length r as

$$r = (1,0,1,1,1,1,1,0,0,1,1,0,1,1,1,0,0,0,1)_2; \quad \text{and}$$
$$r = (1,1,3,3,0,3,1,3,0,1)_4.$$

We proceed down the bits of r_2 and the "bits" of r_4 in the table below, showing the Miller doubling functions and the Miller quadrupling functions respectively. We also give the (symbolic) contributions of the addition stage through each iteration, where the contributions for the quadrupling in the last column come from (4.5).

r_2	doubling function $\ell_{R,R}$	add	r_4	quadrupling function $\ell^* = \ell_{R,R}^2 \cdot \ell_{[2]R,[2]R}$	add
1			1		
0	$1787365x_Q + 3265249y_Q + 1298802$	-		$2193611x_Q + 2615989x_Q^2 + 1272939y_Q$	
1	$499051x_Q + 1519758y_Q + 252180$	$\ell_{R,P}$	1	$+976306x_Qy_Q + 1544654$	$\ell_{R,P}$
1	$1964409x_Q + 4449539y_Q + 3965085$	$\ell_{R,P}$		$472857x_Q + 4631799x_Q^2 + 855294y_Q$	$f_{3,P}\cdot$
1	$1602765x_Q + 4481394y_Q + 3129213$	$\ell_{R,P}$	3	$+863905x_Qy_Q + 4329792$	$\ell_{R,[3]P}$
1	$4488506x_Q + 3612376y_Q + 616267$	$\ell_{R,P}$		$1995376x_Q + 1654387x_Q^2 + 52066y_Q$	$f_{3,P}\cdot$
1	$950057x_Q + 415353y_Q + 4134551$	$\ell_{R,P}$	3	$+1700464x_Qy_Q + 2170487$	$\ell_{R,[3]P}$
0	$2929277x_Q + 4073952y_Q + 2048154$	-		$1312938x_Q + 2295780x_Q^2 + 2324490y_Q$	
0	$395667x_Q + 4338943y_Q + 863219$	-	0	$+1110956x_Qy_Q + 623172$	-
1	$716051x_Q + 2547016y_Q + 158022$	$\ell_{R,P}$		$1222627x_Q + 2354396x_Q^2 + 2809176y_Q$	$f_{3,P}\cdot$
1	$2800109x_Q + 2621579y_Q + 3877545$	$\ell_{R,P}$	3	$+1926011x_Qy_Q + 4044376$	$\ell_{R,[3]P}$
0	$2094414x_Q + 2653532y_Q + 150580$	-		$1776372x_Q + 4473739x_Q^2 + 561014y_Q$	
1	$4235065x_Q + 3824071y_Q + 1409140$	$\ell_{R,P}$	1	$+1818712x_Qy_Q + 4081367$	$\ell_{R,P}$
1	$276753x_Q + 570647y_Q + 3756147$	$\ell_{R,P}$		$598941x_Q + 1564006x_Q^2 + 4147402y_Q$	$f_{3,P}\cdot$
1	$3651399x_Q + 2015011y_Q + 4515851$	$\ell_{R,P}$	3	$+1932170x_Qy_Q + 105642$	$\ell_{R,[3]P}$
0	$4185903x_Q + 4555071y_Q + 3061710$	-		$3586623x_Q + 285761x_Q^2 + 1024273y_Q$	
0	$4506674x_Q + 4492823y_Q + 49462$	-	0	$+4257178x_Qy_Q + 4037012$	-
0	$4095434x_Q + 4138617y_Q + 310038$	-		$4248739x_Q + 867348x_Q^2 + 3404187y_Q$	
1	$2393164x_Q + 2721643y_Q + 2713439$	$\ell_{R,P}$	1	$+1011335x_Qy_Q + 338067$	$\ell_{R,P}$

Table 4.1: Miller functions in the double-and-add routine (left) and the quadruple-and-add routine (right) for a small Tate pairing.

We now give cost estimates for both scenarios. In order to compare the two, we use the ratios of field operations in the speed-record paper by Aranha *et al.* [AKL+11, Table 1]. They found Karatsuba multiplication most successful for both the degree 2 and 3 extensions in their tower, which for each such layer in the tower contributes a factor of 3 and 6 to the complexity of multiplications in $\mathbb{F}_{q^k}$ respectively (refer to Section 2.6.3). In our case $k = 24 = 2^3 \cdot 3$, meaning that $\mathbb{F}_{q^{24}}$ multiplications $\mathbf{m}_k$ cost roughly $3^3 \cdot 6 = 162$ multiplications in $\mathbb{F}_q$, i.e. $\mathbf{m}_k = 162\mathbf{m}_1$. We also follow their $\mathbf{s} - \mathbf{m}$ ratio, and assume squarings in $\mathbb{F}_{q^{24}}$ are roughly 2/3 as expensive as multiplications in $\mathbb{F}_{q^{24}}$, meaning $\mathbf{s}_k = 108\mathbf{m}_1$.

When linear functions are used to update the pairing value, the evaluation of these functions at the coordinates of Q gives a sparse element in $\mathbb{F}_{q^k}$. Optimised routines will take advantage of this sparseness, so we account for this in Table 4.2, where the line functions in the double-and-add routine are always sparse, whereas only some of the addition updates in the quadruple-and-add routine are sparse. We assume a sparse multiplication in $\mathbb{F}_{q^{24}}$ costs around 13/18's that of a full multiplication, by adapting the ratios (modular reductions included) of Aranha *et al.* [AKL+11, Table 1] with the fact that lines in the Tate pairing are slightly more sparse than lines in the ate pairing.

	double-and-add			quadruple-and-add		
operation	#	cost each	total	#	cost each	total
Miller functions	18	$3\mathbf{m}_1 + 5\mathbf{s}_1$	$54\mathbf{m}_1 + 90\mathbf{s}_1$	9	$14\mathbf{m}_1 + 16\mathbf{s}_1$	$126\mathbf{m}_1 + 144\mathbf{s}_1$
evaluation at Q	18	$8\mathbf{m}_1$	$144\mathbf{m}_1$	9	$16\mathbf{m}_1$	$144\mathbf{m}_1$
squarings in $\mathbb{F}_{q^{24}}$	18	$\mathbf{s}_k$	$18\mathbf{s}_k$	18	$\mathbf{s}_k$	$18\mathbf{s}_k$
sparse updates	18	$13/18\mathbf{m}_k$	$13\mathbf{m}_k$	-	-	-
full updates	-	-	-	9	$\mathbf{m}_k$	$9\mathbf{m}_k$
add functions	11	$14\mathbf{m}_1 + 2\mathbf{s}_1$	$154\mathbf{m}_1 + 22\mathbf{s}_1$	7	$14\mathbf{m}_1 + 2\mathbf{s}_1$	$98\mathbf{m}_1 + 14\mathbf{s}_1$
sparse updates	11	$13/18\mathbf{m}_k$	$8\mathbf{m}_k$	7	$13/18\mathbf{m}_k$	$5\mathbf{m}_k$
full updates	-	-	-	4	$\mathbf{m}_k$	$4\mathbf{m}_k$
			$352\mathbf{m}_1 + 112\mathbf{s}_1$ $+21\mathbf{m}_k + 18\mathbf{s}_k$ $\approx 5811\mathbf{m}_1$			$368\mathbf{m}_1 + 158\mathbf{s}_1$ $+18\mathbf{m}_k + 18\mathbf{s}_k$ $\approx 5386\mathbf{m}_1$

Table 4.2: Comparing estimated costs of the double-and-add and quadruple-and-add routines for the Miller loop in a small Tate pairing.

In this case we see that the quadruple-and-add algorithm saves around 3 multiplications in $\mathbb{F}_{q^k}$ at the forfeit of around 60 extra multiplications in $\mathbb{F}_q$. Since $\mathbf{m}_k : \mathbf{m}_1 \approx 162 : 1$, the quadruple-and-add algorithm becomes favoured for this Tate pairing computation. Of course, this analysis does not include the non-negligible cost of field additions, but this favours both sides at different stages; there is clearly a higher number of additions in the computation of the Miller functions for the quadruple-and-add algorithm, but on the other hand the extra full extension field multiplications in the double-and-add algorithm also come incur many additions (again, see [AKL+11, Table 1] for the $k = 12$ ratio). An optimised scheduling to compute the sequence of operations (with a minimal number of additions) in Section 4.2.1 would surely see the quadruple-and-add algorithm as the (albeit slightly) more efficient route to computing $f_{r,P}(Q)$ in the Tate pairing. Finally, one should always keep in mind that the relative speed-ups for optimisations in the Miller loop is always significantly diluted by

the fixed cost of the final exponentiation.

4.3 Octuple-and-add

For octupling, we begin by setting $n = 3$ in (4.2) to obtain the Miller update ℓ^* corresponding the octupling of T as

$$\ell^* = \prod_{i=1}^{3} (g_{[2^{i-1}]T,[2^{i-1}]T})^{2^{3-i}} = \big(g_{T,T}\big)^4 \cdot \big(g_{[2]T,[2]T}\big)^2 \cdot \big(g_{[4]T,[4]T}\big),$$

which has divisor $8(T) - ([8]T) - 7(\mathcal{O})$. We now concentrate individually on the same curve models as in the quadrupling.

4.3.1 Octuple-and-add on $y^2 = x^3 + b$

For the octupling line product, we use homogeneous projective coordinates to give ℓ^* as

$$\begin{aligned}\ell^* = \alpha \cdot (&\ell_{4,0} \cdot x_P^4 + \ell_{3,0} \cdot x_P^3 + \ell_{2,0} \cdot x_P^2 + \ell_{1,0} \cdot x_P \\ &+ \ell_{3,1} \cdot x_P^3 y_P + \ell_{2,1} \cdot x_P^2 y_P + \ell_{1,1} \cdot x_P y_P + \ell_{0,0}),\end{aligned}$$

where α is again contained in a proper subfield of $\mathbb{F}_{p^k}$ and can be eliminated to give $\hat{\ell}^* = \ell^*/\alpha$. The $\ell_{i,j}$ coefficients are

$$\begin{aligned}&\ell_{4,0} = (-9X_1^2 Z_1^2) \cdot S_{4,0}; \quad \ell_{3,0} = (-12 Z_1^2 Y_1^2) \cdot S_{3,0}; \quad \ell_{2,0} = (-54 X_1 Y_1^2 Z_1) \cdot S_{2,0}; \\ &\ell_{1,0} = (-36 X_1^2 Y_1^2) \cdot S_{1,0}; \quad \ell_{0,0} = ((Y_1^2 + 3bZ_1^2) Y_1^2) \cdot S_{0,0}; \quad \ell_{3,1} = (8 Y_1 Z_1^3) \cdot S_{3,1}; \\ &\ell_{2,1} = (216 X_1 Y_1 Z_1^2) \cdot S_{2,1}; \quad \ell_{1,1} = (72 X_1^2 Y_1 Z_1) \cdot S_{1,1}; \quad \ell_{0,1} = (8 Y_1^3 Z_1) \cdot S_{0,1}.\end{aligned}$$

with

$$S_{i,j} = \sum_{k=0}^{11} c_{i,j,k} \cdot (Y_1^2)^{11-k} (bZ_1^2)^k,$$

where $c_{i,j,k}$ is the coefficient of $(Y_1^2)^{11-k}(bZ_1^2)^k$ belonging to $\ell_{i,j}$ (see [CBNW10a, App. B] for the full formulas and Magma scripts). As an example, we have

$$\begin{aligned}\ell_{0,0} = (Y_1^2(Y_1^2 + 3bZ_1^2)) \cdot \big(&Y_1^{22} - 3375bY_1^{20}Z_1^2 - 262449b^2Y_1^{18}Z_1^4 - 2583657b^3Y_1^{16}Z_1^6 \\ &+ 47678058b^4Y_1^{14}Z_1^8 - 40968342b^5Y_1^{12}Z_1^{10} - 272740770b^6Y_1^{10}Z_1^{12} \\ &+ 738702990b^7Y_1^8Z_1^{14} - 669084219b^8Y_1^6Z_1^{16} + 206730549b^9Y_1^4Z_1^{18} \\ &- 23914845b^{10}Y_1^2Z_1^{20} + 14348907b^{11}Z_1^{22}\big).\end{aligned}$$

4.3.2 Octuple-and-add on $y^2 = x^3 + ax$

Following the trend of the fastest formulas for the $n = 1$ and $n = 2$ cases for curves of this shape, we again projectify ℓ^* using weight-$(1, 2)$ coordinates to give

$$\begin{aligned}\ell^* = \alpha \cdot (&\ell_{4,0} \cdot x_P^4 + \ell_{3,0} \cdot x_P^3 + \ell_{2,0} \cdot x_P^2 + \ell_{1,0} \cdot x_P \\ &+ \ell_{3,1} \cdot x_P^3 y_P + \ell_{2,1} \cdot x_P^2 y_P + \ell_{1,1} \cdot x_P y_P + \ell_{0,0}),\end{aligned}$$

where we ignore the subfield cofactor α to give $\hat{\ell}^* = \ell^*/\alpha$. The $\ell_{i,j}$ coefficients are given as

$$\begin{aligned}&\ell_{4,0} = (-4X_1^2Z_1^4) \cdot S_{4,0}, \quad \ell_{3,0} = (-16X_1^3Z_1^3) \cdot S_{3,0}, \quad \ell_{2,0} = (-8X_1^4Z_1^2) \cdot S_{2,0} \\ &\ell_{1,0} = (16X_1^5Z_1) \cdot S_{1,0}, \quad \ell_{0,0} = (4X_1^6) \cdot S_{0,0}, \quad \ell_{3,1} = (4Y_1Z_1^4) \cdot S_{3,1} \\ &\ell_{2,1} = (4X_1Y_1Z_1^3) \cdot S_{2,1}, \quad \ell_{1,1} = (4X_1^2Y_1Z_1^2) \cdot S_{1,1}, \quad \ell_{0,1} = (4X_1^3Y_1Z_1) \cdot S_{0,1},\end{aligned}$$

with

$$S_{i,j} = \sum_{k=0}^{16} c_{i,j,k} \cdot (X_1^2)^{16-k}(aZ_1^2)^k,$$

where $c_{i,j,k}$ is the coefficient of $(X_1^2)^{16-k}(aZ_1^2)^k$ belonging to $\ell_{i,j}$ (see [CBNW10a, App. B] for the full formulas). As an example, we have

$$\begin{aligned}\ell_{2,0} = -8X1^4Z1^2 \cdot \big(&189X_1^{32} + 882aX_1^{30}Z_1^2 + 6174a^2X_1^{28}Z_1^4 - 26274a^3X_1^{26}Z_1^6 \\ &- 1052730a^4X_1^{24}Z_1^8 - 449598a^5X_1^{22}Z_1^{10} - 1280286a^6X_1^{20}Z_1^{12} - 1838850a^7X_1^{18}Z_1^{14} \\ &- 23063794a^8X_1^{16}Z_1^{16} - 1543290a^9X_1^{14}Z_1^{18} + 539634a^{10}X_1^{12}Z_1^{20} + 646922a^{11}X_1^{10}Z_1^{22} \\ &+ 1386918a^{12}X_1^8Z_1^{24} + 75846a^{13}X_1^6Z_1^{26} + 17262a^{14}X_1^4Z_1^{28} + 922a^{15}X_1^2Z_1^{30} - 35a^{16}Z_1^{32}\big).\end{aligned}$$

Remark 4.3.1 (Why Tate and not ate?). The method of 2^n-tupling described so far in this chapter can achieve a slight speed-up for Tate-like pairings with large embedding degrees. However, unfortunately for the case of loop unrolling, at these higher security levels (where ate-like pairings reign supreme over Tate-like pairings), our method of merging is not preferable in the ate setting. In Tate-like pairings, the elliptic curve operations are performed in $\mathbb{F}_q$, whilst in ate-like pairings they are performed in $\mathbb{F}_{q^{k/d}}$, $d \leq 6$. The fact that $d \leq 6$ means that the ratio between $\mathbb{F}_{q^k}$ multiplications and $\mathbb{F}_{q^{k/d}}$ multiplications is no more than 18, which is not large enough to make the merging trade-offs favourable. On the other hand, in our Tate pairing example with $k = 24$, we saw that the analogous ratio was 162, which certainly presented a case for the larger window size.

4.4 Fixed Argument Pairings

This section targets the very common scenario that arises in many pairing-based protocols where one argument in the pairing is fixed as a long term secret key or a constant parameter in the system. We present solid speed-ups in both the Tate and ate settings by extending prior methods of pairing precomputation according to the techniques described in the previous sections of this chapter.

For ease of exposition, we will assume that we are back in the ate pairing setting, so that our pairing is $e(Q, P) = f_{T,Q}(P)^{(q^k-1)/r}$. Note that although we write T for the loop length, if the ate pairing is not optimal (i.e. $T > \log_2 r/\varphi(k)$), then the application of the techniques in this chapter to an optimal variant is identical. We simply replace T with an appropriately $m \approx \log_2 r/\varphi(k)$ in Algorithm 2.3. The important point to note is that, unlike the previous sections in this chapter, Algorithm 2.3 for the (optimal) ate pairing computes point operations and Miller functions depending on the first argument Q, which is defined over the full extension field $\mathbb{F}_{q^k}$.

If both Q and P are fixed, then the pairing $e(Q, P)$ itself can be precomputed and stored. If the second argument $P = (x_P, y_P)$ is the one that is fixed, this does not facilitate advantageous precomputations since all that Algorithm 2.3 needs from P is its coordinates x_P and y_P in the evaluation stages (Steps 5 and 9). If however, it is the first argument Q that is fixed, then precomputations become very useful. Scott [Sco05a] was the first to point out that the first argument (in our case Q) being fixed allows us to precompute all of its multiples $R = [v]Q$ that

occur in the Miller loop. We explore Scott's observation a little further by posing the question: *what is the best we can do with Q to optimally alleviate the cost of the pairing once P arrives?* We work under the assumption that (within reason) we are not too concerned with the cost of any precomputations. If Q is a long term secret key that is fixed for many runs of a protocol over days/months/years, then it is reasonable to assume that even if precomputations were several orders of magnitude more costly than a pairing, a user would still take this route if it facilitated significant speed-ups over many future encryptions or decryptions. Besides, these precomputations take place "offline", and in any case the precomputations we employ will not be orders of magnitude more expensive than any single pairing: if a stand-alone pairing takes less than a millisecond [AKL+11], then the precomputation phase we suggest will take at most a few milliseconds.

4.4.1 Merging Miller functions (properly!)

We start by slightly tweaking Scott's initial suggestion [Sco05a] to precompute the multiples of the point Q in Miller's algorithm, i.e. all of the intermediate points $R = (x_R, y_R)$ in Algorithm 2.3. Instead of storing R at each stage, we will instead store the line functions $\ell_{R,R}$ and $\ell_{R,P}$ that arise. Suppose for now we are working in affine coordinates, and let the j-th (doubling or addition) Miller line function that arises be $\ell_j : y - (\lambda_j x + \nu_j) = 0$. Taking advantage of Q being fixed, we can compute all such λ_j, ν_j offline and store all of the Miller functions as $\mathcal{L} = [(\lambda_1, \nu_1), ..., (\lambda_n, \nu_n)]$. Algorithm 2.3 for the pairing computation then simplifies to Algorithm 4.2 below.

Algorithm 4.2 Miller's algorithm for a fixed argument ate pairing.

Input: $P \in \mathbb{G}_1$, $\mathcal{L} = [(\lambda_1, \nu_1), ..., (\lambda_n, \nu_n)]$, and $T = (T_{n-1} \dots T_1 T_0)_2$ with $T_{n-1} = 1$.
Output: $f_{T,Q}(P)^{(q^k-1)/r} \leftarrow f$.

1: $j \leftarrow 1$, $f \leftarrow 1$.
2: **for** $i = n - 2$ down to 0 **do**
3: $(\lambda, \nu) \leftarrow \mathcal{L}[j]$, $j \leftarrow j + 1$.
4: $f \leftarrow f^2 \cdot (y_P - (\lambda x_P + \nu))$.
5: **if** $r_i = 1$ **then**
6: $(\lambda, \nu) \leftarrow \mathcal{L}[j]$, $j \leftarrow j + 1$.
7: $f \leftarrow f \cdot (y_P - (\lambda x_P + \nu))$.
8: **end if**
9: **end for**
10: **return** $f \leftarrow f^{(q^k-1)/r}$.

Note that $\mathcal{L}$ contains both the doubling and additions lines in the order that they arise in the algorithm. Further note that working in affine coordinates facilitates lines of the form $\ell : y-(\lambda x+\nu)$, whilst working in projective space gives lines which are of the form $\ell : \ell_{0,1}y + \ell_{1,0}x + \ell_{0,0}$. The former is slightly preferred as its evaluation at $P = (x_P, y_P)$ at runtime only requires one multiplication (λ by x_P), rather than the two that would occur in the latter ($\ell_{0,1}$ by y_P and $\ell_{1,0}$ by x_P). In this case then, we are happy to put up with the more expensive (affine) operations in the precomputation phase because it facilitates a faster pairing at runtime.

The key observation in understanding our improvement is the following. In the previous sections we merged iterations to trade full extension field operations for many subfield operations. For larger embedding degrees where the ratio between the complexities of these operations grows large, this trade off becomes slightly favourable (at least in the context of the Tate pairing). In the case of a fixed argument pairing though, the subfield operations used to merge the iterations can all be done in the precomputation phase. What was a trade-off now becomes much more of a one-sided affair, and now we manage to save costly multiplications in $\mathbb{F}_{q^k}$ for the small price of evaluating (but no longer computing!) pairing functions with a few more terms.

However, the approach we describe in this section significantly improves the original approach in [CS10]. Namely, adopting projective coordinates gives much faster formulas than those that were initially suggested which arise from using affine coordinates. For example, observe the difference between the merged update functions in both scenarios:

$$\begin{aligned}
\text{affine update}: \quad & \ell_{4,0}x_P^4 + \ell_{3,0}x_P^3 + \ell_{2,0}x_P^2 + \ell_{1,0}x_P + \ell_{0,0} \\
& \quad + x_P^3 y_P + \ell_{2,1}x_P^2 y_P + \ell_{1,1}x_P y_P + \ell_{0,1}y_P = 0; \\
\text{projective update}: \quad & \ell_{2,0}x_P^2 + \ell_{1,0}x_P + \ell_{0,0} + \ell_{1,1}x_P y_P + \ell_{0,1}y_P = 0.
\end{aligned}$$

Notice the coefficient of $x_P^3 y_P$ being 1, i.e. $\ell_{3,1} = 1$. The affine update was taken according to the general expression in [CS10, Lemma 1, Eq. 3], whilst the projective equation comes from our derivations in Section 4.2.1 and Section 4.2.2. The reason the projective version is much simpler is partly because deriving the formulas for ℓ^* as a function of one point allows for Gröbner basis-like simplifications between its coordinates, but more so because this derivation si-

multaneously facilitated big denominator eliminations that reduced the degree of ℓ^* – refer back to the α's in 4.2.1 and 4.2.2. In addition, one more observation that aids our cause is that we can divide the projective update freely to eliminate one of the coefficients, e.g. dividing the projective update above by $\ell_{2,0}$ gives

$$\ell^* : x_P^2 + \ell'_{1,0}x_P + \ell'_{0,0} + \ell'_{1,1}x_P y_P + \ell'_{0,1}y_P = 0,$$

where $\ell'_{i,j} = \ell_{i,j}/\ell_{2,0}$. Of course, we could have also done this in the original (non-merged) case, i.e. divided $\ell_{0,1}y + \ell_{1,0}x + \ell_{0,0}$ through by $\ell_{0,1}$ to give $y + \ell'_{1,0}x + \ell'_{0,0}$. Thus, in the projective case we can also benefit from one of the coefficients being 1. It is only in the original (non-merged) case that the line functions resulting from the affine and projective derivation techniques turn out the same. As soon as merging is advantageous, then projective coordinates perform far better in the fixed argument scenario.

4.4.2 Example: the record-holding BN curve

We give an example of the proposed method on the BN curve that was found by Nogami *et al.* [NAS+08], and soon after used to break the software speed-record by Aranha *et al.* [AKL+11]. Referring back to the parameterisations for the BN family from Example 2.5.6 or Example 2.6.7, Nogami *et al.* found that $x = -(2^{62} + 2^{55} + 1)$ gives $q(x)$ and $r(x)$ as 254-bit primes, making this a prime choice for pairings at the 128-bit security level. Recall from Example 2.6.7 that an optimal ate pairing on BN curves is achieved with the loop parameter for Miller's algorithm as $6x + 2$. In the case of the chosen curve then, the 65-bit binary representation of our loop parameter (we can take $-(6x + 2)$ to make it positive – see Section 2.6.4) becomes

$$6 \cdot (2^{62} + 2^{55} + 1) + 2 = (1, 1, 0, 0, 0, 0, 0, 1, 1, \underbrace{0, \quad \ldots, 0, \ldots \quad, 0}_{53 \text{ zeros}}, 1, 0, 0)_2.$$

We will compare the double-and-add, quadruple-and-add, and octuple-and-add versions of Algorithm 4.2. The line function updates in the three scenarios be-

come:

$$\begin{aligned}
\text{doubling} - \S3.3.1: \quad \ell^*_{\text{dbl}}(P): \quad & y_P + \ell_{1,0}x_P + \ell_{0,0} = 0; \qquad (4.6)\\
\text{quadrupling} - \S4.2.1: \quad \ell^*_{\text{qrpl}}(P): \quad & x_P^2 + \ell_{1,0}x_P + \ell_{1,1}x_Py_P + \ell_{0,1}y_P + \ell_{0,0} = 0;\\
\text{octupling} - \S4.3.1: \quad \ell^*_{\text{octpl}}(P): \quad & x_P^4 + \ell_{3,0}x_P^3 + \ell_{2,0}x_P^2 + \ell_{1,0}x_P + \ell_{3,1}x_P^3y_P\\
& + \ell_{2,1}x_P^2y_P + \ell_{1,1}x_Py_P + \ell_{0,1}y_P + \ell_{0,0} = 0,
\end{aligned}$$

where we have divided through by the leftmost $\ell_{i,j}$ in each case to make one of the coefficients 1. We can treat each of the $x_P^iy_P^j$ terms as lying in $\mathbb{F}_q$ (more on this in a moment), so the cost of evaluating each ℓ^* at P is found by enumerating the multiplications between $\ell_{i,j} \in \mathbb{F}_{q^2}$ and $x_P^iy_P^j \in \mathbb{F}_q$, which each cost $2\mathbf{m}_1$. Thus, the cost of the function evaluations in doubling is $2\mathbf{m}_1$, in quadrupling is $3 \times 2\mathbf{m}_1 = 6\mathbf{m}_1$, and in octupling is $7 \times 2\mathbf{m}_1 = 14\mathbf{m}_1$. The values $\ell_{i,j}$ are all in $\mathbb{F}_{q^2}$ because we are employing a sextic twist to compute them in $\mathbb{F}_{q^{k/6}}$.

We write the base-2, base-4 and base-8 representations of $6x + 2$ as

$$\begin{aligned}
6x + 2 &= (1,1,0,0,0,0,0,1,1,\underbrace{0,\quad \ldots,0,\ldots\quad,0}_{53\text{ zeros}},1,0,0)_2 && \text{(65 bits)};\\
&= (1,2,0,0,3,\underbrace{0,\quad \ldots,0,\ldots\quad,0}_{26\text{ zeros}},1,0)_4 && \text{(33 digits)};\\
&= (3,0,1,4,\underbrace{0,\quad \ldots,0,\ldots\quad,0}_{17\text{ zeros}},4)_8 && \text{(22 digits)}.
\end{aligned}$$

We now focus on the cost of additions that will occur in each scenario. Recall the function update in the addition phase from Equation (4.5), which rewritten in our (ate pairing) context is

$$\ell^+ = f_{w,Q}(P) \cdot \ell_{R,[w]Q}(P),$$

where $0 < w < 2^n$ is the digit occurring in the base-2^n representation of the loop parameter. Note that we can precompute $[w]Q$ for all of the w that appear in the corresponding representation, and therefore we can also precompute $f_{w,Q}$. In cases where it is convenient, we can use the optimised doubling or quadrupling formulas to simplify these computations. For example, in the octupling we encounter non-zero digits 1 (once), 3 (once) and 4 (twice). Thus, we will need to compute $f_{1,Q} \cdot \ell_{R,Q} = \ell_{R,Q}$ (a standard addition), $f_{3,Q} \cdot \ell_{R,[3]Q}$ and $f_{4,Q} \cdot \ell_{R,[4]Q}$. The computation of $f_{4,Q}$ can be done with the quadrupling formulas (and once

evaluated at P, used in both iterations where it occurs), whilst $f_{3,Q} = f_{2,Q} \cdot \ell_{Q,[2]Q}$ can be computed with the doubling formulas followed by a line computation.

A very careful analysis of the operations involved in each scenario gives the counts detailed in Table 4.3.

	op.	double-and-add (no merging) #	cost	quadruple-and-add (merging 2 iters.) #	cost	octuple-and-add (merging 3 iters.) #	cost
tuple	eval. $\ell^*(P)$	64	$128\mathbf{m}_1$	32	$192\mathbf{m}_1$	21	$294\mathbf{m}_1$
	$f \leftarrow f^2$	63	$63\mathbf{s}_k$	63	$63\mathbf{s}_k$	64	$64\mathbf{s}_k$
	$f \leftarrow f \cdot \ell^*(P)$	63	$63\tilde{\mathbf{m}}_k$	32	$32\mathbf{m}_k$	21	$21\mathbf{m}_k$
add	eval. $\ell_{R,Q}(P)$	4	$8\mathbf{m}_1$	3	$18\mathbf{m}_1$	4	$56\mathbf{m}_1$
	$f \leftarrow f \cdot \ell_{R,[w]Q}(P)$	4	$4\tilde{\mathbf{m}}_k$	3	$3\tilde{\mathbf{m}}_k$	3	$3\tilde{\mathbf{m}}_k$
	eval. $f_{w,Q}(P)$	-	-	2	$6\mathbf{m}_1$	2	$10\mathbf{m}_1$
	$f \leftarrow f \cdot f_{w,Q}(P)$	-	-	2	$3\tilde{\mathbf{m}}_k$	2	$2\mathbf{m}_k + \hat{\mathbf{m}}_k$
total			$\approx 5017\mathbf{m}_1$		$\approx 4446\mathbf{m}_1$		$\approx 4053\mathbf{m}_1$

Table 4.3: Operation counts for the Miller loop of a fixed argument pairing on a BN curve employing our merging technique for precomputation vs. the previous method of precomputation (no merging).

We have distinguished sparse multiplications $\tilde{\mathbf{m}}_k$ and sparser multiplications $\hat{\mathbf{m}}_k$ from full multiplications $\mathbf{m}_k$, where sparse multiplications indicate a line function multiplied by a full element in $\mathbb{F}_{q^k}$, and sparser multiplications mean a multiplication between two line functions, as in [AKL+11]. We use the ratios $\tilde{\mathbf{m}}_k \approx 13/18\mathbf{m}_k$ and $\hat{\mathbf{m}}_k \approx 5/9\mathbf{m}_k$, taken conservatively from [AKL+11, Table 1.], from which we also take $\mathbf{m}_k = \mathbf{m}_{12} \approx 54\mathbf{m}_1$ and $\mathbf{s}_k \approx 2/3\mathbf{m}_k \approx 36\mathbf{m}_1$.

To finish the example, we return to our statement that we can treat each of the $x_P^i y_P^j$ terms as base field elements, so that their multiplication by $\ell_{i,j} \in \mathbb{F}_{q^2}$ counts as $2\mathbf{m}_1$. This is indeed the case, but the details need to be discussed diligently. Suppose we are using the same tower chosen in [AKL+11]:

$$\mathbb{F}_q \xrightarrow{u^2+1} \mathbb{F}_{q^2} \xrightarrow{v^3-(u+1)} \mathbb{F}_{q^6} \xrightarrow{w^2-v} \mathbb{F}_{q^{12}}.$$

The curve used is $E/\mathbb{F}_q : y^2 = x^3 + 2$ and the correct sextic can be chosen as $E/\mathbb{F}_{q^2} : y^2 = x^3 + 2/(u+1)$. In this case, we have $w^6 = (u+1)$ so that our twisting isomorphism is $\Psi^{-1} : E \to E'$, $(x, y) \mapsto (x/w^2, y/w^3)$, and our untwisting isomorphism is $\Psi : E' \to E$, $(x', y') \mapsto (x'w^2, y'w^3)$. We rewrite Ψ^{-1} multiplicatively as $(x, y) \mapsto (x(1-u)v^2/2, y(1-u)vw/2)$. Thus, if we are applying Theorem 3.1 to compute entirely on the twist, we will actually map P to

$P' = (x'_P, y'_P)$ via Ψ^{-1}, and evaluate the functions there (rather than untwisting Q), i.e. we will be multiplying the $\ell_{i,j} \in \mathbb{F}_{q^2}$ by x'_P, y'_P. The coordinate $x'_P = x_P(1-u)v^2/2$, where $x_P \in \mathbb{F}_q$, and similarly $y'_P = y_P(1-u)vw/2$ where $y_P \in \mathbb{F}_q$. Thus, each of $x'^i_P y'^j_P$ terms from (4.6) can be thought of as $x_P y_P \in \mathbb{F}_q$ with some product of u, v, w attached; this product simply determines the placement of $x_P y_P$ in the structure that represents $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$. As an example, we will write $x'^3_P y'_P$ that arises in the octupling as $x'^3_P y'_P = x_P y_P \cdot ((1-u)v^2/2)^3(1-u)vw/2$, from which we need to sort out the mess that is attached to $x_P y_P$. It becomes $\frac{(1-u)^4 v^7 w}{16}$, where $(1-u)^4$ simplifies to -4, and v^7 simplifies to $2uv$, giving $x'_P y'_P = -(x_P y_P)/2 \cdot uvw$, which allows us to easily place $x'^3_P y'_P$ (and multiples of it) in the structure for $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$.

4.4.3 Example: a $k = 24$ BLS curve

We take a look at another example which targets the 256-bit security level. Once again we use a $k = 24$ curve from the BLS family, the parameterisations of which can be seen in Example 2.6.9 or Example 2.6.11. A nice choice of x at this level is $x = -(2^{23} + 2^{42} + 2^{44} + 2^{64})$ (taken from our recent work [CLN11, Table 7]), for which q is 639 bits and r is 513 bits. The curve is $E/\mathbb{F}_q : y^2 = x^3 - 2$, and a nice choice for the extension field tower is

$$\mathbb{F}_q \xrightarrow{u^2+1} \mathbb{F}_{q^2} \xrightarrow{v^2-(u+1)} \mathbb{F}_{q^4} \xrightarrow{w^3-v} \mathbb{F}_{q^{12}} \xrightarrow{z^2-w} \mathbb{F}_{q^{24}}.$$

The twisted curve is $E'/\mathbb{F}_{q^4} : y^2 = x^3 - 2/v$, so in the ate pairing we perform the elliptic curve and Miller function computations in $\mathbb{F}_{q^4}$. In this case the ate pairing is already optimal (see Example 2.6.9), so the Miller function is computed as $f_{x,Q}(P)$. The update functions are of the same form as the last example, i.e. as they are in (4.6).

We write the base-2, base-4 and base-8 representations of x (which coinci-

dentally have the same bit lengths as the last example) as

$$\begin{aligned} x &= (1, \underbrace{0, \ \ldots \ , 0}_{19 \text{ zeros}}, 1, 0, 1, \underbrace{0, \ \ldots \ , 0}_{18 \text{ zeros}}, 1, \underbrace{0, \ \ldots \ , 0}_{23 \text{ zeros}}) && \text{(65 bits);} \\ &= (1, \underbrace{0, \ \ldots \ , 0}_{9 \text{ zeros}}, 1, 1, \underbrace{0, \ \ldots \ , 0}_{9 \text{ zeros}}, 2, \underbrace{0, \ \ldots \ , 0}_{11 \text{ zeros}})_4 && \text{(33 bits);} \\ &= (2, \underbrace{0, \ \ldots \ , 0}_{6 \text{ zeros}}, 5, \underbrace{0, \ \ldots \ , 0}_{6 \text{ zeros}}, 4, \underbrace{0, \ \ldots \ , 0}_{7 \text{ zeros}})_8 && \text{(22 bits).} \end{aligned}$$

Once again, when 2 is encountered in the base-4 representation for quadrupling, the addition update is $f_{2,Q}(P) \cdot \ell_{R,[2]Q}(P)$, and so is the product of two lines ($f_{2,Q}$ and $\ell_{R,[2]Q}$ are precomputed). In the octupling, we precompute $f_{4,Q}$ using quadrupling and further precompute $f_{5,Q}$ as $f_{5,Q} = f_{4,Q} \cdot \ell_{[4]Q,Q}$ (but keep these components separate). The first 2 encountered in the octupling require the computation of $f_{2,P}$ only. When we come across 4 in the octupling, the update is computed by evaluating $f_{4,Q}(P)$ and $\ell_{R,[4]Q}(P)$ and then taking the product; similarly, when we come across 5 we evaluate $f_{4,Q}(P)$, $\ell_{[4]Q,Q}$ and $\ell_{R,[5]Q}(P)$ separately before taking their product, noting that this incurs two sparse multiplications.

We carry over all the same ratios used in the previous example, since the nature of the sextic twist is the same. The tower has one extra layer of degree 2, so there is another factor of 3 introduced in extension field multiplications, i.e. $\mathbf{m}_k \approx 162\mathbf{m}_1$, and each multiplication of an $\ell_{i,j}$ by $x_P y_P$ (or $x'_P y'_P$) incurs $4\mathbf{m}_1$. We give careful estimates in Table 4.4.

		double-and-add (no merging)		quadruple-and-add (merging 2 iters.)		octuple-and-add (merging 3 iters.)	
	op.	#	cost	#	cost	#	cost
tuple	eval. $\ell^*(P)$	64	$256\mathbf{m}_1$	32	$384\mathbf{m}_1$	21	$588\mathbf{m}_1$
	$f \leftarrow f^2$	63	$63\mathbf{s}_k$	63	$63\mathbf{s}_k$	64	$64\mathbf{s}_k$
	$f \leftarrow f \cdot \ell^*(P)$	63	$63\tilde{\mathbf{m}}_k$	32	$32\mathbf{m}_k$	21	$21\mathbf{m}_k$
add	eval. $\ell_{R,Q}(P)$	3	$12\mathbf{m}_1$	3	$36\mathbf{m}_1$	3	$84\mathbf{m}_1$
	$f \leftarrow f \cdot \ell_{R,[w]Q}(P)$	3	$3\tilde{\mathbf{m}}_k$	3	$2\tilde{\mathbf{m}}_k + \hat{\mathbf{m}}_k$	3	$3\tilde{\mathbf{m}}_k$
	eval. $f_{w,Q}(P)$	-	-	1	$4\mathbf{m}_1$	1	$12\mathbf{m}_1$
	$f \leftarrow f \cdot f_{w,Q}(P)$	-	-	1	$\mathbf{m}_k$	2	$2\mathbf{m}_k$
total			$\approx 14794\mathbf{m}_1$		$\approx 12898\mathbf{m}_1$		$\approx 11673\mathbf{m}_1$

Table 4.4: Operation counts for the Miller loop of a fixed argument pairing on a BLS curve employing our merging technique for precomputation vs the previous method of precomputation (no merging).

4.4.4 Storage requirements and applications

We summarise the results from the examples in Section 4.4.2 and Section 4.4.3 in Table 4.5. To estimate the cost of the Miller loop that employs no precomputation, we used the numbers from our optimised formulas in Table 3.2. We took the added cost of each doubling step as $2\mathbf{m} + 7\mathbf{s} \approx 7\mathbf{m}$ and each addition step as $3\mathbf{m} + 8\mathbf{s} \approx 9\mathbf{m}$. These operations take place in $\mathbb{F}_{q^2}$ for the BN curve and in $\mathbb{F}_{q^4}$ for the BLS curve, where multiplications cost $3\mathbf{m}_1$ and $9\mathbf{m}_1$ respectively. Calculating the storage requirement in each scenario is straightforward. For example, in the standard (doubling) case on a BN curve, we have to store 64 doubling lines and 4 addition lines of the form $\ell : y + \ell_{1,0}x + \ell_{0,0}$, so we need to store $68 \cdot 2$ values in $\mathbb{F}_{q^2}$, or $68 \cdot 2 \cdot 2$ 254-bit elements, which totals $68 \cdot 2 \cdot 2 \cdot 254 \approx 70,000$ bits. As another example, for the octupling on a BLS curve, we have to store roughly 22 doubling updates which each require the storage of 8 elements $\ell_{i,j} \in \mathbb{F}_{q^4}$, plus 3 addition updates of roughly the same size, giving $25 \cdot 8 \cdot 4 \cdot 639 \approx 510,000$ bits.

	128-bit optimal pairing $k = 12$ BN curve $\mathbb{F}_q = 254$ bits		256-bit optimal pairing $k = 24$ BLS curve $\mathbb{F}_q = 639$ bits	
precomp method	Miller loop cost	$\approx$ storage required (bits)	Miller loop cost	$\approx$ storage required (bits)
none	6469 $\mathbf{m}_1$	-	19069 $\mathbf{m}_1$	-
standard [Sco05a]	5017 $\mathbf{m}_1$	70,000	14794 $\mathbf{m}_1$	340,000
quadrupling	4446 $\mathbf{m}_1$	75,000	12898 $\mathbf{m}_1$	368,000
octupling	4053 $\mathbf{m}_1$	100,000	11673 $\mathbf{m}_1$	510,000

Table 4.5: Computational savings in the Miller loop vs. storage costs for fixed argument pairings employing our merging technique.

We remark that these methods are obviously not intended for resource or storage constrained environments. However, taking advantage of precomputation in the fixed argument scenario is clearly very advantageous if one can afford the storage. As we mentioned earlier, the reader should keep in mind that relative savings within the Miller loop are always diluted when considered as relative savings across the entire pairing computation, due to the fixed cost of the final exponentiation that does not benefit from any precomputations. In this light, our estimated ratios between "no precomputation" and "standard precomputation" techniques agree with Scott's recent timings [Sco11, Table 2] on 128-bit BN curves, if we keep in mind that the Miller loop and final exponentiation are

roughly the same cost at this level. In that paper Scott discusses a wide range of protocol specific optimisations beyond (but including) the fixed argument optimisation, where he mentions that our technique would provide a useful speedup at twice the storage cost [Sco11, §5]. However, this comment was based on the prior, but inferior technique in [CS10]. Indeed, the original quadrupling functions were as big as our modified octupling functions, so Scott's comment about the storage requirements was certainly valid. In our updated version though, we can see that the storage requirements are not greatly affected by stepping up to quadrupling, since we essentially store half the functions of twice the size.

There are many pairing-based cryptosystems that can benefit from precomputation when one of the arguments is fixed. In Table 4.6, we have listed some pairing-based cryptosystems that have fixed arguments and hence which can benefit from the improvements in this chapter.

	# pairings	fixed arguments	# pairings	fixed arguments
Public key encryption	**Encryption**		**Decryption**	
Boyen-Mei-Waters [BMW05]	0		1	2^{nd}
ID-based encryption	**Encryption**		**Decryption**	
Boneh-Franklin [BF03]	1	2^{nd}	1	1^{st}
Boneh-Boyen [BB11]	0		1	2^{nd}
Waters [Wat05]	0		2	both in 2^{nd}
Attribute-based encr.	**Encryption**		**Decryption**	
GPSW [GPSW06, §4]	0		$\leq$ #attr.	all in 1^{st}
LOSTW [LOS$^+$10, §2]	0		$\leq 2 \cdot$ #attr.	all in 2^{nd}
ID-based signatures	**Signing**		**Verification**	
Waters [Wat05]	0		2	1 in 2^{nd}
ID-based key exchange	**Initiator**		**Responder**	
Smart-1 [Sma02]	2	1 in 1^{st}, 1 in 2^{nd}	2	1 in 1^{st} , 1 in 2^{nd}
Chen-Kudla [CK03]	1	1^{st}	1	2^{nd}
McCullagh-Barreto [MB05]	1	2^{nd}	1	2^{nd}

Table 4.6: Fixed arguments in various pairing-based cryptosystems.

In some cases, such as the Boneh-Franklin identity-based encryption scheme [BF03] or the Chen-Kudla identity-based key agreement protocol [CK03], one party computes a pairing where the first argument is fixed while the other party computes a pairing where the second argument is fixed; here, our technique can only be applied to speed up one party's pairing computation.

In others cases, such as the McCullagh-Barreto identity based key agreement protocol [MB05], the two parties employing the cryptosystem can both benefit because they each compute pairings where the fixed value appears in the same

argument of the pairing. Our speed-up is also applicable to attribute-based encryption schemes such as those of Goyal *et al.* [GPSW06, §4] and Lewko *et al.* [LOS+10, §2] which perform a large number of pairings (one or two for each attribute used in decryption), as in these schemes the second arguments in all these pairing computations are long-term values (fixed across multiple runs of the protocol, though not all identical within a single run of the protocol). Again, we refer to Scott's recent work for more details concerning other special improvements in these scenarios [Sco11].

4.5 Summary of contributions

In Sections 4.1-4.3 of this chapter, we looked at loop unrolling in Miller's algorithm. We showed that merging two successive iterations can be preferable for Tate pairing computations, but further discussed in Remark 4.3.1 that merging is not preferable for one-off ate pairings. In Section 4.4 however, we applied the loop unrolling technique to the common scenario of fixed argument pairings. When considering the runtime of the Miller loop, we showed that our (modified) method facilitates speed-ups of around 20% over the previous method of precomputation. If storage space permits, then taking advantage of a fixed argument can give a speed-up of 35% over a Miller loop that does not use precomputation.

Both of our examples in this chapter focussed on curves with sextic twists and smooth embedding degrees belonging to families that are the front-runners for their respective target security levels. We expect that the speed up for other families would be even larger, since if a sextic twist isn't used the relative size of the field where the precomputation savings take place grows, i.e. the subfield operations have a greater overall effect on the complexity of the Miller loop. Even in the case of say, a quartic twist, our method is likely to give better relative savings than those reported here. This is also due to the Miller lines in the traditional double-and-add version being "not as sparse" in the case of lower degree twists, meaning that the comparison between multiplication complexities favours our case even more.

Finally, we remark that if works like that of Aranha *et al.* [AKMRH11] see a resurgence of the Weil pairing $e(P,Q) = f_{m,P}(Q)/f_{m,Q}P$ in some scenarios, then this could further aid the case for loop unrolling in state-of-the-art implementations.

Chapter 5

Attractive subfamilies of BLS curves for high-security pairings

In this chapter and the next, we address the same problem: finding subfamilies of parameterised curve families that offer certain advantages with respect to their associated pairing implementation.

Here we consider the Barreto-Lynn-Scott (BLS) $k = 24$ family [BLS02] (see also [FST10, §6.6]) in detail, since these curves are a stand-out candidate for implementing high-security pairings. In the next chapter we apply similar techniques to all of the other attractive families of curves.

Current public-key security recommendations have influenced a concentrated effort from the pairing-based community towards optimising the implementation of pairings on Barreto-Naehrig (BN) curves [BN05]. Indeed, aside from an array of many other attractive properties (refer back to Section 4.4.2, Example 2.6.7, or see [Nae09, PJNB11] for more details), BN curves are perfectly suited to the security level of 128 bits, since they achieve an optimal balance between the necessary sizes of the three groups involved in the pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ (see Figure 2.39). The BN pairing speed record of 10 million cycles set by Hankerson *et al.* in 2008 [HMS08] stood until mid 2010, when three papers appeared in rapid succession [NNS10, BGDM+10, AKL+11], each one shaving more time off the previous record and pushing the limit of efficiency at this security level. This work was pinnacled by Aranha *et al.* [AKL+11], who applied a combination of improvements to accelerate the entire pairing computation to less than 2 million

cycles. As implementors seemingly converge towards a "satisfaction asymptote" at the 128-bit security level, the focus is now beginning to shift to optimising pairings at higher security levels.

The BLS family has already been identified as the prime candidate for 256-bit secure pairings by Scott [Sco11], who now holds the current software speed record at this level. The aim of our work is to provide some of the finer details that will begin to pave the way for implementors who may wish to further accelerate the state-of-the-art timings on BLS curves. Much of our discussion is motivated by Pereira *et al.*'s work in the case of BN curves [PJNB11], where they detail a very simple method of generating highly optimal instantiations of *implementation-friendly* BN curves. With the same intent, we point out four highly attractive subfamilies of BLS curves that facilitate very efficient instantiations of high-security pairings.

The proposed curves are found by restricting the search parameter x in the polynomial representation to any one of four specific congruency classes, namely $x_0 \equiv 7, 16, 31, 64 \bmod 72$. These choices result in prime fields of characteristic $q \equiv 19 \bmod 24$, which in turn leads to three very efficient towering options for the full extension field $\mathbb{F}_{q^{24}}$. We show that all curves found in any of the proposed subfamilies can immediately be given by the same short Weierstrass equation over $\mathbb{F}_q$, and the unique sextic twist E' of correct order for use in the ate pairing setting is automatically determined. Not only is there no computational effort necessary to write down the curve equations once the prime q is found, both E and E' can be represented very compactly by the polynomial parameter x_0 alone.

We give some details on line function computation that make an important link to Theorem 3.1. As a resource for implementors, we give elaborate lists of example curves covering a range of high-security levels between 192- and 320-bit security, including the dedicated BLS security level of 256 bits. We have chosen the examples with a very sparse parameter x which leads to a low number of addition steps in the Miller loop and simultaneously very efficient exponentiations by x, which are needed in the final exponentiation. We have already shown that the ate pairing for this family is already optimal in Example 2.6.9, and provided details on an efficient version of the final exponentiation routine in Example 2.6.11. The example curves we provide can all be found in Appendix A.

The remainder of this chapter is organised as follows. We briefly recall the polynomials for all the necessary parameters below, before describing the pro-

posed implementation-friendly subfamilies of BLS curves in Section 5.1. Section 5.2 details some choices that can facilitate simpler pairing code. Finally, in Section 5.3 we provide timing results of a C implementation of the optimal ate pairing on BLS curves.

The BLS parameterisations. We recall the BLS parameterisations from any of Examples 2.5.5, 2.6.9, or 2.6.11, but add some details that are necessary for this work:

$$\begin{aligned} q(x) &= (x-1)^2(x^8-x^4+1)/3+x; \qquad r(x) = x^8-x^4+1; \qquad t(x) = x+1; \\ n(x) &= (x-1)^2(x^8-x^4+1)/3; \qquad f(x) = (x-1)(2x^4-1)/3. \end{aligned} \tag{5.1}$$

Finding a specific BLS curve is achieved by running through integer values $x_0 \equiv 1 \bmod 3$ until $q(x_0)$ and $r(x_0)$ are both prime (note that $x_0 \equiv 1 \bmod 3$ leads to all involved parameters being integers). For each set of parameters, there exists an elliptic curve E over $\mathbb{F}_q$ such that $\#E(\mathbb{F}_q) = n(x_0)$. The correct curve E can be found by trying different values for b (i.e. different twists) and checking for the right group order. Another alternative is to compute the coefficient by the algorithm described in [RS10]. Since $r(x_0) \mid n(x_0)$, there is a subgroup of $E(\mathbb{F}_q)$ of prime order $r(x_0)$. The CM discriminant is $D = 3$ because $4q(x_0) - t(x_0)^2 = 3f(x_0)^2$. The family has a ρ-value of $\rho = \deg(q)/\deg(r) = 1.25$.

5.1 Particularly friendly subfamilies

In this section we show that specialising the congruency classes of the curve-finding search parameter gives rise to four subfamilies of $k = 24$ BLS curves that are highly efficient in terms of all operations required in a pairing computation. Specifically, we show that rather than searching with $x_0 \equiv 1 \bmod 3$, searching with any of $x_0 \equiv 7, 16, 31, 64 \bmod 72$ guarantees that the curves found offer (among other things) the following advantages.

- **The curve constant b is immediately determined** (see Proposition 5.4 below). This saves performing extra computations that test different values of b and the corresponding group order until the correct twist is found; or the checks for quadratic and cubic residuosity and root computations for the algorithm in [RS10].

x_0 mod 72	$q(x_0)$ mod 72 *(eq. (5.1))*	$n(x_0)$ mod 72 *(eq. (5.1))*	efficient tower *(Prop. 5.3)*	E *(Prop. 5.4)*	E' *(Prop. 5.5)*
7	19	12	✓	$y^2 = x^3 + 1$	$y^2 = x^3 \pm 1/v$
16	19	3	✓	$y^2 = x^3 + 4$	$y^2 = x^3 \pm 4v$
31	43	12	✓	$y^2 = x^3 + 1$	$y^2 = x^3 \pm v$
64	19	27	✓	$y^2 = x^3 - 2$	$y^2 = x^3 \pm 2/v$

Table 5.1: Four attractive subfamilies of BLS curves.

– **Highly efficient field tower options are available** (see Proposition 5.3 below). This facilitates very efficient field arithmetic in the full extension field (and all intermediate subfields).

– **The correct twist is immediately determined** (see Proposition 5.5 below). Among other savings, having an automated and general representation for the *correct* twist saves group arithmetic on curves over the quartic extension field $\mathbb{F}_{q^4}$ in the generation phase. Also, the representations of the twists are always simple and facilitate nice back-and-forth isomorphisms between E and E'.

In addition to the above efficiency benefits, generating curves with identical or consistent parameters also offers the advantage of code reusability across different instantiations and security levels. This allows an implementor the flexibility of scaling parameter sizes to better match a neighbouring security level without changing any of the pairing code.

Furthermore, having fixed coefficients for the curve and its twist leads to a very compact way of representing the curve data. Note that in this setting, the knowledge of the generating parameter x_0 uniquely determines all information about both curves. What only remains is to give generators for the groups $\mathbb{G}_1$ and $\mathbb{G}_2'$, except for the case of $x_0 \equiv 64 \bmod 72$ for which a compact generator in $\mathbb{G}_1$ is always available as $[h](3, 5)$, where h is the cofactor which maps elements of $E(\mathbb{F}_p)$ into $\mathbb{G}_1$.

We split the rest of this section into two subsections. The first subsection is dedicated to proving the claims in Table 5.1 and showing that taking $x_0 \equiv 7, 16, 31, 64 \bmod 72$ will always give rise to highly efficient BLS instantiations.

The intention of the second subsection is to detail why $x_0 \equiv 7, 16, 31, 64 \bmod 72$ reign supreme over the other possible congruence classes.

5.1.1 Using the four classes $x_0 \equiv 7, 16, 31, 64 \bmod 72$

We start with a lemma that is instrumental in some of the proofs that follow. For a prime q, let $\mathrm{QR}(q)$ denote the set of quadratic residues modulo q.

Lemma 5.1. *Let $x_0 \in \mathbb{Z}$ be any of $x_0 \equiv 7, 16, 31, 64 \bmod 72$, and let $q = q(x_0)$ with q given by (5.1) be a prime. Then 2 is neither a quadratic nor a cubic residue modulo q.*

Proof. A simple calculation shows that for $x_0 \equiv 7, 16, 64 \bmod 72$, we have $q(x_0) \equiv 19 \bmod 72$. For $x_0 \equiv 31 \bmod 64$, we have $q(x_0) \equiv 43 \bmod 72$. In both cases, it is easy to deduce that $2 \notin \mathrm{QR}(q)$ by computing the Legendre symbol.

It remains to show that 2 is not a cube modulo q for each of the x_0 values. For all four cases, we use [IR90, Prop. 9.6.2], which states that for $q \equiv 1 \bmod 3$, 2 is a cubic residue modulo q if and only if there exist integers C, D such that $q = C^2 + 27D^2$. According to [IR90, Prop. 8.3.2], for $q \equiv 1 \bmod 3$ there always exist integers A and B such that $4q = A^2 + 27B^2$ and A, B are unique up to sign. Thus, if A and B in this unique representation are both even, 2 is a cube modulo q, otherwise it is not. In each of the four cases $x_0 \equiv 7, 16, 31, 64 \bmod 72$, we examine A and B in terms of the polynomials from (5.1). The CM norm equation for the BLS family is $4q = t^2 + 3f^2$, thus we obtain f in terms of the polynomial representation as $f(x) = (x-1)/3 \cdot (2x^4 - 1)$.

For $x_0 \equiv 64 \bmod 72$, we have $f(x_0) \equiv 0 \bmod 3$ which allows us to write $4q(x_0) = t(x_0)^2 + 27 \cdot (f(x_0)/3)^2$, where $A = t(x_0) = x_0 + 1 \equiv 65 \bmod 72$ and $B = f(x_0)/3 = (x_0 - 1)/9 \cdot (2x_0^4 - 1) \equiv 25 \bmod 72$ are both odd, meaning that for $x_0 \equiv 64 \bmod 72$, 2 is not a cubic residue modulo $q(x_0)$.

For the other three cases $x_0 \equiv 7, 16, 31 \bmod 72$, it is easy to show that $f(x_0) \not\equiv 0 \bmod 3$, so the CM equation does not directly yield A and B. The two cases $x_0 \equiv 7, 16 \bmod 72$ can be handled by considering a transformation of the CM norm as

$$4q = \left(\frac{3f + t}{2}\right)^2 + 27\left(\frac{t - f}{6}\right)^2.$$

For $x_0 \equiv 7 \bmod 72$, $t(x_0), f(x_0) \equiv 2 \bmod 6$, so that both $A = (3f + t)/2$ and $B = (t - f)/6$ are integers. Furthemore, $f(x_0) \equiv 2 \mod 4$ and $t(x_0) \equiv 0 \bmod 4$

reveal that $3f + t \equiv 2 \bmod 4$ so that A is odd, from which it follows that 2 is not a cubic residue modulo $q(x_0)$ for $x_0 \equiv 7 \bmod 72$.

For $x_0 \equiv 16 \bmod 72$, we have $t(x_0), f(x_0) \equiv 1 \bmod 2$ and $t(x_0), f(x_0) \equiv 2 \bmod 3$. Furthermore, since $t(x_0) \equiv 1 \bmod 4$ and $f(x_0) \equiv 3 \bmod 4$, again we conclude that $3f + t \equiv 2 \bmod 4$, meaning that $A = (t - 3f)/2$ is odd and 2 is not a cube modulo $q(x_0)$ for $x_0 \equiv 16 \bmod 72$.

Finally, for $x_0 \equiv 31 \bmod 72$, we require a slightly different transformation of the CM equation as

$$4q = \left(\frac{t - 3f}{2}\right)^2 + 27\left(\frac{t + f}{6}\right)^2.$$

In this case $t(x_0) \equiv 2 \bmod 6$ and $f(x_0) \equiv 4 \bmod 6$ so that $A = (t - 3f)/2$ and $B = (t+f)/6$ are integers. Since $t(x_0) \equiv 0 \bmod 4$ and $f(x_0) \equiv 2 \bmod 4$, it follows that $A \equiv 1 \bmod 4$ is odd and 2 is not cube modulo $q(x_0)$ for $x_0 \equiv 31 \bmod 72$. □

The ideal way to form a quadratic extension field arises when $q \equiv 3 \bmod 4$ which allows us to take $\mathbb{F}_{q^2} = \mathbb{F}_q(u),\ u^2 + 1 = 0$. Operations in $\mathbb{F}_q[u]/(u^2 + 1)$ are cheaper than operations in $\mathbb{F}_q[u]/(u^2 - \alpha)$ for any other non-residue $\alpha \in \mathbb{F}_q$, since multiplication by $\alpha \neq \pm 1$ costs additions in $\mathbb{F}_q$ (cf. [DOSD06]). Since we have $q \equiv 19 \bmod 24$ in all four cases, we always have $q \equiv 3 \bmod 4$ and can take advantage of this extension.

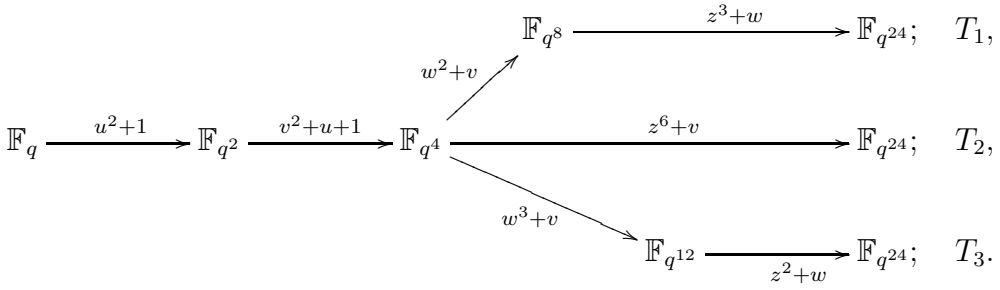
For the extension from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}(v)$, the ideal irreducible binomial in terms of simplicity would be $v^2 + u$. Unfortunately the following proposition shows that if we form $\mathbb{F}_{q^2}$ as above, then this binomial cannot be used to define $\mathbb{F}_{q^4}$ (and this statement is true for any quartic extension fields, whether in the context of pairings or not).

Proposition 5.2. *If $q \equiv 3 \bmod 4$, and $\mathbb{F}_{q^2}$ is constructed as $\mathbb{F}_q(u),\ u^2 + 1 = 0$, then the polynomial $x^2 + su$ with $s \in \mathbb{F}_q$ is reducible over $\mathbb{F}_{q^2}$. In particular, $\mathbb{F}_{q^4}$ cannot be constructed over $\mathbb{F}_{q^2}$ using a binomial of the above form.*

Proof. Since $-1 \notin \mathrm{QR}(q)$, precisely one of $s/2$ or $-s/2$ is a quadratic residue modulo q. In the first case, take $s = 2a^2$ for some $a \in \mathbb{F}_q$ and write $x^2 + su = x^2 + 2a^2u = (x + au - a)(x - au + a)$. In the second case, taking $s = -2a^2$ for some $a \in \mathbb{F}_q$ gives $x^2 + su = x^2 - 2a^2u = (x + au + a)(x - au - a)$. Thus, $x^2 + su$ with $s \in \mathbb{F}_q$ is reducible over $\mathbb{F}_{q^2}$. □

Nevertheless, a binomial that is almost as attractive in terms of efficiency is $v^2 + (u + 1)$, since multiplications by $u + 1$ in $\mathbb{F}_{q^2}$ also come almost for free. The following proposition shows that the proposed subfamilies of BLS curves always allow $\mathbb{F}_{q^4}$ to be constructed using this binomial. Furthermore, we also show that the rest of the tower up to $\mathbb{F}_{q^{24}}$ can be constructed in three different ways; all three of which employ optimal binomials, but may be preferred by implementors depending on various factors, such as the nature of previous pairing code, or whether compression is desired.

Proposition 5.3. *Let $x_0 \in \mathbb{Z}$ be any of $x_0 \equiv 7, 16, 31, 64 \bmod 72$. If $q = q(x_0)$ given by the polynomial in (5.1) is prime, then the extension field $\mathbb{F}_{q^{24}}$ can be constructed using any of the following towering options T_1, T_2, T_3:*



Proof. For all four congruency classes $x_0 \equiv 7, 16, 31, 64 \bmod 72$, we have $q(x_0) \equiv 19 \bmod 24$, so that $q \equiv 3 \bmod 4$ and we can use $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 1)$. For the remaining irreducibility arguments, we make use of Theorem C.1, which is due to Benger and Scott. We immediately note that $2, 3 \mid q - 1$.

Let us compute $N_{\mathbb{F}_{q^2}/\mathbb{F}_q}(-(u+1)) = N_{\mathbb{F}_{q^2}/\mathbb{F}_q}(-1)N_{\mathbb{F}_{q^2}/\mathbb{F}_q}(u+1) = (u+1)^{q+1} = (u+1)^q(u+1) = (1-u)(u+1) = 2$. Since 2 is not a quadratic residue modulo q, Theorem C.1 ensures that $v^2 + u + 1$ is irreducible in $\mathbb{F}_{q^2}[v]$ and we can construct $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[v]/(v^2 + u + 1)$.

Next, we compute $N_{\mathbb{F}_{q^4}/\mathbb{F}_q}(-v) = N_{\mathbb{F}_{q^4}/\mathbb{F}_q}(-1)N_{\mathbb{F}_{q^4}/\mathbb{F}_q}(v) = N_{\mathbb{F}_{q^4}/\mathbb{F}_q}(v) = v^{1+q+q^2+q^3} = (v^{1+q^2})^{1+q} = ((-(u+1))^{1+q^2})^{(1+q)/2} = ((u+1)^2)^{(1+q)/2} = (u+1)^{q+1} = 2$. Lemma 5.1 tells us that 2 is neither a quadratic nor a cubic residue modulo q, and thus it follows from Theorem C.1 that $w^2 + v$, $w^3 + v$ and $z^6 + v$ are all irreducible over $\mathbb{F}_{q^4}$, giving rise to the tower T_2, to T_1 up to $\mathbb{F}_{q^8}$ and T_3 up to $\mathbb{F}_{q^{12}}$.

For the remaining parts of T_1 and T_3, we similarly compute $N_{\mathbb{F}_{q^{12}}/\mathbb{F}_q}(-w) = 2^{(1+q^4+q^8)/3}$ and $N_{\mathbb{F}_{q^8}/\mathbb{F}_q}(-w) = 2^{(1+q^4)/2}$. Since $q \equiv 1 \bmod 6$ for all cases, it follows that $(1 + q^4 + q^8)/3$ is odd and so $N_{\mathbb{F}_{q^{12}}/\mathbb{F}_q}(-w)$ is not a square in $\mathbb{F}_q$

since 2 is not a square by Lemma 5.1. Similarly, $(q^4+1)/2 \equiv 1 \bmod 3$ and so $N_{\mathbb{F}_{q^8}/\mathbb{F}_q}(-w)$ is not a cube in $\mathbb{F}_q$. Therefore, Theorem C.1 ensures irreducibility of the remaining polynomials and completes the proof. □

We have now shown that once a BLS prime q is found using $x_0 \equiv 7, 16, 31, 64 \bmod 72$, a highly efficient tower is immediately available. The following two propositions show that the curve constant and twisted curve are also immediate in all four cases.

Proposition 5.4. *If $x_0 \equiv 7, 31 \bmod 72$ in (5.1) produces a prime $q = q(x_0)$, then the curve $E/\mathbb{F}_q : y^2 = x^3 + 1$ is always such that $r = r(x_0) \mid n = \#E(\mathbb{F}_q)$. Similarly, for $x_0 \equiv 16 \bmod 72$, the desired curve is always $E/\mathbb{F}_q : y^2 = x^3 + 4$. Finally, for $x_0 \equiv 64 \bmod 72$, the desired curve is always $E/\mathbb{F}_q : y^2 = x^3 - 2$.*

Proof. It is well known that if g is neither a square nor a cube in $\mathbb{F}_q$, then all possible group orders an elliptic curve $E : y^2 = x^3 + b$ can have over $\mathbb{F}_q$ occur as the order of one of the 6 twists with $b \in \{1, g, g^2, g^3, g^4, g^5\}$. Specifically, choosing b as exactly one of $\{1, g, g^2, g^3, g^4, g^5\}$ will give the correct number of points (cf. [Sil09, §X.5]), i.e. the curve with $r \mid n = \#E(\mathbb{F}_q)$. Lemma 5.1 shows that we can take $g = 2$, so that in all four cases the correct b is exactly one of $\{1, 2, 4, 8, 16, 32\}$.

For both $x_0 \equiv 7 \bmod 72$ and $x_0 \equiv 31 \bmod 72$, we have $n(x_0) = (x_0 - 1)^2(x_0^8 - x_0^4 + 1)/3 \equiv 12 \bmod 72$ from (5.1). Thus, both cases have $2, 3 \mid n$, meaning that the correct curves $E/\mathbb{F}_q$ necessarily contain points of order 2 and points of order 3. This implies that b is both a quadratic and cubic residue modulo q, from which it follows that $b = 1$ is the only option.

For $x_0 \equiv 16 \bmod 72$, observe that $n(x_0) \equiv 3 \bmod 72$ and thus the correct curve E has a point of order 3, but not a point of order 2. This rules out $b = 1, 8$, since the points $(-1, 0)$ and $(-2, 0)$ have order 2 on the respective curves. The curve $E/\mathbb{F}_q : y^2 = x^3 + b$ has a point of order 3 if and only if b is a square in $\mathbb{F}_q$, which rules out $b = 2$, and therefore $b = 32$ as well. To rule out $b = 16$, we first observe that since $n = n(x_0) \equiv 3 \bmod 72$, $9 \nmid n$ and $E/\mathbb{F}_q$ has at most three points of order 3. If $b = 16$, then two such points are $(0, -4)$ and $(0, 4)$. It is easy to see that $-3 \in \mathrm{QR}(q)$, so let $\nu^2 = -3$ for $\nu \in \mathbb{F}_q$, and write $P = (-4, 4\nu) \in E(\mathbb{F}_q) : y^2 = x^3 + 16$. An easy calculation (e.g. using point doubling formulas) shows that $[2]P = (-4, -4\nu) = -P$, so that P has order 3, and similarly for $-P$. Thus, there are at least four points of order 3 in $\mathbb{F}_q$ if $b = 16$ contradicting $9 \nmid n$, which leaves $b = 4$ as the only option.

For $x_0 \equiv 64 \bmod 72$, we can make use of Algorithm 3.5 in [RS10], where in our case $U = t/2$ and $V = f/2$. Since $2V = f(x_0) \equiv 0 \bmod 3$ and $2U = t(x_0) = 2 \bmod 3$, we immediately have $E/\mathbb{F}_q : y^2 = x^3 + 16$ as the correct curve. Lastly, since $-2 \in \mathrm{QR}(q)$, write $\mu^2 = -2$ for $\mu \in \mathbb{F}_q$, so that $\mu^6 = -8$. Since $E/\mathbb{F}_q : y^2 = x^3 + 16$ is isomorphic to $\tilde{E}/\mathbb{F}_q : y^2 = x^3 + 16/\mu^6$ over $\mathbb{F}_q$, we can take $b = 16/-8 = -2$ as the curve constant instead. □

Proposition 5.5. *If $x_0 \equiv 7 \bmod 72$ produces the BLS curve $y^2 = x^3+1$ described in Proposition 5.4, and $\mathbb{F}_{q^4}$ is constructed as in Proposition 5.3, then the correct sextic twist with $r = r(x_0) \mid \#E'(\mathbb{F}_{q^4})$ can be obtained as both $E'/\mathbb{F}_{q^4} : y^2 = x^3 + 1/v$ and $E'/\mathbb{F}_{q^4} : y^2 = x^3 - 1/v$. Similarly, $x_0 \equiv 16 \bmod 72$ gives rise to the correct twist as both $E'/\mathbb{F}_{q^4} : y^2 = x^3 + 4v$ or $E'/\mathbb{F}_{q^4} : y^2 = x^3 - 4v$; $x_0 \equiv 31 \bmod 72$ gives rise to the correct twist as both $E'/\mathbb{F}_{q^4} : y^2 = x^3 + v$ or $E'/\mathbb{F}_{q^4} : y^2 = x^3 - v$; and finally, $x_0 \equiv 64 \bmod 72$ gives rise to the correct twist as both $E'/\mathbb{F}_{q^4} : y^2 = x^3 + 2v$ or $E'/\mathbb{F}_{q^4} : y^2 = x^3 - 2v$.*

Proof. We first note that $-1 = u^6$ is a sixth power in $\mathbb{F}_{q^2}$. Therefore, for $b' \in \mathbb{F}_{q^4}$, the curves given by $y^2 = x^3 + b'$ and $y^2 = x^3 - b'$ are isomorphic over $\mathbb{F}_{q^4}$.

In each case the correct twist E' is unique with the property that $r \mid \#E'(\mathbb{F}_{q^4})$ and it has degree 6 (see [HSV06]). Since there are exactly two twists of E of degree 6 over $\mathbb{F}_{q^4}$, there are only two possible group orders. We have $\#E(\mathbb{F}_{q^4}) = q^4 + 1 - t_4$, where t_4 can be computed from t and f (notation as before) as $t_4 = (t^4 - 18t^2f^2 + 9f^2)/8$ (see [Sil09, §V.2]). If we define f_4 by $4q^4 = t_4^2 + 3f_4^2$, the possible group orders for the sextic twist are given in [HSV06] as

$$\begin{aligned} n_{4,1} &= q^4 + 1 - (3f_4 - t_4)/2, \\ n_{4,2} &= q^4 + 1 - (-3f_4 - t_4)/2. \end{aligned}$$

We compute both $n_{4,1}$ and $n_{4,2}$ as polynomials in terms of the parameterisation (5.1), and evaluate them at each of the congruency classes which reveals opposing parities each time. The remainder of the proof is essentially the same for all four congruency classes, so we demonstrate completely with $x_0 \equiv 16 \bmod 72$. Taking $x_0 \equiv 16 \bmod 72$ gives $n_{4,1} \equiv 28 \bmod 72$ and $n_{4,2} \equiv 49 \bmod 72$. In particular, $n_{4,1}$ is even and $n_{4,2}$ is odd in this case. From the polynomial parameterisation, it is easy to check that $r \mid n_{4,1}$. Therefore, the unique sextic twist we are looking for in this case has an even group order over $\mathbb{F}_{q^4}$.

Proposition 5.3 shows that v is neither a square nor a cube in $\mathbb{F}_{q^4}$, so the

correct twist can be given as $y^2 = x^3 + 4v$ or as $y^2 = x^3 + 4v^5$. We have $4v = (N_{\mathbb{F}_{q^4}/\mathbb{F}_q}(v))^2 v = v^{3+2q+2q^2+2q^3}$ as in the proof of Proposition 5.3. Since the exponent on the right hand side of the last equation is divisible by 3, we conclude that $4v$ is a cube in $\mathbb{F}_{q^4}$. Similarly, one can show that $4v^5$ is not a cube. Since $4v$ is a cube, the curve $y^2 = x^3 + 4v$ has a point of order 2, namely $(-c, 0)$ with $c^3 = 4v$. Hence its order is even and we have found the correct twist for $x_0 \equiv 16 \bmod 72$. The other three cases are proved analogously. □

5.1.2 The other congruence classes

We now show why the four congruence classes $x_0 \equiv 7, 16, 31, 64 \bmod 72$ stand out over the other congruency classes. After all, restricting $x_0 \equiv 1 \bmod 3$ to any or all of the proposed classes essentially discards 20 out of 24 other congruency classes modulo 72. Of course, there will always be examples where some of the discarded congruency classes produce curves that also perform highly efficiently. However, we argue that $x_0 \equiv 7, 16, 31, 64 \bmod 72$ are the only classes for which we can *always* simultaneously guarantee the propositions in the previous subsection.

Quadratic extension to $\mathbb{F}_{q^2}$

We start by eliminating the classes of $x_0 \bmod 72$ which do not facilitate the quadratic extension as $\mathbb{F}_{q^2} = \mathbb{F}_q(u),\ u^2 + 1 = 0$. Of the 24 possible x_0 values modulo 72 in $\{1, 4, 7, ..., 67, 70\}$, 12 have $x_0 \equiv 1, 10 \bmod 12$, which always (undesirably) produce $q \equiv 1 \bmod 12$. The remaining 12 are $\{4, 7, 16, 19, 28, 31, 40, 43, 52, 55, 64, 67\}$ with $x_0 = 4, 7 \bmod 12$, which always produce $q(x_0) \equiv 7 \bmod 12$.

Quadratic extension to $\mathbb{F}_{q^4}$

If $\mathbb{F}_{q^4}$ is to be constructed as $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}(v),\ v^2 + u + 1 = 0$, then (refer back to the proof of Proposition 5.3) we can only guarantee this if $N_{\mathbb{F}_{q^2}/\mathbb{F}_q}(-(u+1)) = 2$ is not a quadratic residue in $\mathbb{F}_q$. Substituting the remaining 12 candidates for $x_0 \bmod 72$ into (5.1) reveals only 4 possibilities for $q \bmod 72$, those being $q \equiv 7, 19, 43, 55 \bmod 72$. It is easy to check that only two of these have $2 \notin \mathrm{QR}(q)$, namely $q \equiv 19, 43 \bmod 72$. These correspond to 6 of the remaining x_0 congruences, shrinking the pool of preferred candidates to $\{7, 16, 31, 40, 55, 64\}$.

Sextic extension to $\mathbb{F}_{q^{24}}$

The proposed sextic extensions in Proposition 5.3 that employ simple binomials to form $\mathbb{F}_{q^{24}}$ over $\mathbb{F}_{q^4}$ require that 2 is not a cube in $\mathbb{F}_q$. This does not always happen for $x_0 = \{40, 55\}$. For the sake of counter examples, $x_0 = 12856 \equiv 40 \bmod 72$ and $x_0 = 1135 \equiv 55 \bmod 72$ produce BLS curves where 2 is a cube modulo q, and therefore fields which can not use the tower in Proposition 5.3. Even if alternative binomials can be found in such cases, the fact that 2 is a cubic residue also affects the ease of guaranteeing the smallest curve constant b, as we were able to do for the proposed four congruency classes in Proposition 5.4. For example, the smallest curve constant for the curve found with $x_0 = 12856 \equiv 40 \bmod 72$ is $b = -3$, whilst the smallest constant for the curve found with $x_0 = 25312 \equiv 40 \bmod 72$ is $b = 9$.

5.2 Choosing simple lines: twisting vs. untwisting

The aim of this short section is to detail some choices that can facilitate more simple (and theoretically faster) pairing code.

As we mentioned briefly in the previous section, the choice of which tower (T_1, T_2, T_3 in Proposition 5.3) to use in an implementation could be influenced by a number of factors. For example, if low bandwidth requirements favoured maximum compression techniques [SB04], then T_1 would seem most appropriate. On the other hand, if the major priority is raw speed, then one could employ the field arithmetic presented in [DOSD06, §6] and favor the (slightly faster) quadratic over cubic extension offered by T_3. Or perhaps most commonly, if the implementor is adopting BLS curves to scale a prior (say BN) pairing implementation to a higher security level, then the towered code for the BN sextic extension from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^{12}}$ could be easily updated to BLS code for the extension from $\mathbb{F}_{q^4}$ to $\mathbb{F}_{q^{24}}$. In any case, there are efficient formulas available for all three of the towering choices (see both Chapter 3 and [DOSD06]), so we will treat all three cases in parallel and highlight the differences that arise.

One such difference lies in the sparse doubling and addition lines that are used to update the pairing function. For each of the proposed congruency classes, Table 5.2 follows the exposition in [Sco09, §5] and details the correct placing of

the line function coefficients for the three tower choices T_1, T_2, T_3 and the two twist choices in Proposition 5.5, both of which have the same group order but result in different looking line functions.

The point P is always kept in affine coordinates (x_P, y_P). For the affine formulas the line simply is $\ell(P) = y_P - \lambda x_P - \nu$ (see (2.5) and (2.4)), where $\lambda \in \mathbb{F}_{q^4}$ is the slope of the line as usual and $\nu \in \mathbb{F}_{q^4}$ is the constant coefficient. From Chapter 3, we know that projective formulas for these line functions usually output three coefficients $\ell_{0,0}, \ell_{1,0}, \ell_{0,1} \in \mathbb{F}_{q^4}$ that define the evaluated update $\ell(P) = \ell_{0,0} + \ell_{1,0} x_{\psi^{-1}(P)} + \ell_{0,1} y_{\psi^{-1}(P)} \in \mathbb{F}_{q^{24}}$ by being attached to different algebraic elements in the representation of $\mathbb{F}_{q^{24}}$ over $\mathbb{F}_{q^4}$, depending on the twisting isomorphism ψ. In all cases $\ell(P)$ is a sparse element of $\mathbb{F}_{q^{24}}$, with the only difference being the places that the ℓ_i occupy as a result of the different algebraic towerings and maybe a sign change.

To explain the different lines in Table 5.2, let us briefly look at the conversion between representations of an element in the different towering options T_1, T_2, and T_3. Let us start with an element a in T_2 given by the coefficients $a_0, \ldots, a_5 \in \mathbb{F}_{q^4}$, i.e.

$$a = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4 + a_5 z^5.$$

Converting to T_1, we take $z^3 \mapsto w$ and thus the same element is represented as

$$a = (a_0 + a_3 w) + (a_1 + a_4 w) z + (a_2 + a_5 w) z^2.$$

To go to T_3, we use $z \mapsto -uz$ (i.e. $z^2 \mapsto w$) and obtain

$$a = (a_0 + a_2 w + a_4 w^2) + (-a_1 u - a_3 u w - a_5 u w^2) z.$$

An optimised routine must take advantage of the sparse nature of $\ell(P)$ and tailor make a specialised multiplication routine to exploit the presence of zero entries. Roughly speaking, the options in Table 5.2 will give rise to similar speeds, but it is obvious to see which twist constant an implementor would choose (all other things being equal) if their choice of tower is already concrete. For example, if T_1 is the chosen tower, then using $b' = -bv$ gives a slightly easier line function to code than using $b' = bv$.

A more important difference arises when choosing whether to leave $P \in E$ and

cong. class	twist b'	tower choice	twist isomorphism $\psi^{-1} : E \to E'$	Miller lines: Affine	Miller lines: Projective
16,31	bv	T_1	$(x, y) \mapsto (-z^2x, uwy)$	$[\underset{1}{-\nu} : \underset{w}{y_P \cdot u} \mid \underset{1}{\underline{0}} : \underset{w}{\underline{0}} \mid \underset{1}{\lambda \cdot x_P} : \underset{w}{\underline{0}}]$	$[\underset{1}{\ell_{0,0}} : \underset{w}{\ell_{0,1} \cdot y_P u} \mid \underset{1}{\underline{0}} : \underset{w}{\underline{0}} \mid \underset{1}{-\ell_{1,0} \cdot x_P} : \underset{w}{\underline{0}}]$
		T_2	$(x, y) \mapsto (-z^2x, uz^3y)$	$[\underset{1}{-\nu} : \underset{z}{\underline{0}} : \underset{z^2}{\lambda \cdot x_P} : \underset{z^3}{y_P \cdot u} : \underset{z^4}{\underline{0}} : \underset{z^5}{\underline{0}}]$	$[\underset{1}{\ell_{0,0}} : \underset{z}{\underline{0}} : \underset{z^2}{-\ell_{1,0} \cdot x_P} : \underset{z^3}{\ell_{0,1} \cdot y_P u} : \underset{z^4}{\underline{0}} : \underset{z^5}{\underline{0}}]$
		T_3	$(x, y) \mapsto (-wx, wzy)$	$[\underset{1}{-\nu} : \underset{w}{\lambda \cdot x_P} : \underset{w^2}{\underline{0}} \mid \underset{1}{\underline{0}} : \underset{w}{y_P} : \underset{w^2}{\underline{0}}]$	$[\underset{1}{\ell_{0,0}} : \underset{w}{-\ell_{1,0} \cdot x_P} : \underset{w^2}{\underline{0}} \mid \underset{1}{\underline{0}} : \underset{w}{\ell_{0,1} \cdot y_P} : \underset{w^2}{\underline{0}}]$
	$-bv$	T_1	$(x, y) \mapsto (z^2x, wy)$	$[\underset{1}{-\nu} : \underset{w}{y_P} \mid \underset{1}{\underline{0}} : \underset{w}{\underline{0}} \mid \underset{1}{-\lambda \cdot x_P} : \underset{w}{\underline{0}}]$	$[\underset{1}{\ell_{0,0}} : \underset{w}{\ell_{0,1} \cdot y_P} \mid \underset{1}{\underline{0}} : \underset{w}{\underline{0}} \mid \underset{1}{\ell_{1,0} \cdot x_P} : \underset{w}{\underline{0}}]$
		T_2	$(x, y) \mapsto (z^2x, z^3y)$	$[\underset{1}{-\nu} : \underset{z}{\underline{0}} : \underset{z^2}{-\lambda \cdot x_P} : \underset{z^3}{y_P} : \underset{z^4}{\underline{0}} : \underset{z^5}{\underline{0}}]$	$[\underset{1}{\ell_{0,0}} : \underset{z}{\underline{0}} : \underset{z^2}{\ell_{1,0} \cdot x_P} : \underset{z^3}{\ell_{0,1} \cdot y_P} : \underset{z^4}{\underline{0}} : \underset{z^5}{\underline{0}}]$
		T_3	$(x, y) \mapsto (wx, uwzy)$	$[\underset{1}{-\nu} : \underset{w}{-\lambda \cdot x_P} : \underset{w^2}{\underline{0}} \mid \underset{1}{\underline{0}} : \underset{w}{-y_P \cdot u} : \underset{w^2}{\underline{0}}]$	$[\underset{1}{\ell_{0,0}} : \underset{w}{\ell_{1,0} \cdot x_P} : \underset{w^2}{\underline{0}} \mid \underset{1}{\underline{0}} : \underset{w}{-\ell_{0,1} \cdot y_P u} : \underset{w^2}{\underline{0}}]$
	twist b'	tower choice	untwist isomorphism $\psi : E' \to E$	Miller lines: Affine	Miller lines: Projective
7,64	b/v	T_1	$(x', y') \mapsto (-z^2x', uwy')$	$[\underset{1}{y_P} : \underset{w}{-\nu \cdot u} \mid \underset{1}{-\lambda \cdot x_P \cdot u} : \underset{w}{\underline{0}} \mid \underset{1}{\underline{0}} : \underset{w}{\underline{0}}]$	$[\underset{1}{\ell_{0,1} \cdot y_P} : \underset{w}{\ell_{0,0} \cdot u} \mid \underset{1}{\ell_{1,0} \cdot x_P \cdot u} : \underset{w}{\underline{0}} \mid \underset{1}{\underline{0}} : \underset{w}{\underline{0}}]$
		T_2	$(x', y') \mapsto (-z^2x', uz^3y')$	$[\underset{1}{y_P} : \underset{z}{\lambda \cdot u \cdot x_P} : \underset{z^2}{\underline{0}} : \underset{z^3}{-\nu \cdot u} : \underset{z^4}{\underline{0}} : \underset{z^5}{\underline{0}}]$	$[\underset{1}{\ell_{0,1} \cdot y_P} : \underset{z}{-\ell_{1,0} \cdot u \cdot x_P} : \underset{z^2}{\underline{0}} : \underset{z^3}{\ell_{0,0} \cdot u} : \underset{z^4}{\underline{0}} : \underset{z^5}{\underline{0}}]$
		T_3	$(x', y') \mapsto (-wx', wzy')$	$[\underset{1}{y_P} : \underset{w}{\underline{0}} : \underset{w^2}{\underline{0}} \mid \underset{1}{\lambda \cdot x_P} : \underset{w}{-\nu} : \underset{w^2}{\underline{0}}]$	$[\underset{1}{\ell_{0,1} \cdot y_P} : \underset{w}{\underline{0}} : \underset{w^2}{\underline{0}} \mid \underset{1}{-\ell_{1,0} \cdot x_P} : \underset{w}{\ell_{0,0} \cdot u} : \underset{w^2}{\underline{0}}]$
	$-b/v$	T_1	$(x', y') \mapsto (z^2x', wy')$	$[\underset{1}{y_P} : \underset{w}{-\nu} \mid \underset{1}{\lambda \cdot x_P} : \underset{w}{\underline{0}} \mid \underset{1}{\underline{0}} : \underset{w}{\underline{0}}]$	$[\underset{1}{y_P} : \underset{w}{\ell_{0,0}} \mid \underset{1}{-\ell_{1,0} \cdot x_P} : \underset{w}{\underline{0}} \mid \underset{1}{\underline{0}} : \underset{w}{\underline{0}}]$
		T_2	$(x', y') \mapsto (z^2x', z^3y')$	$[\underset{1}{y_P} : \underset{z}{-\lambda \cdot x_P} : \underset{z^2}{\underline{0}} : \underset{z^3}{-\nu} : \underset{z^4}{\underline{0}} : \underset{z^5}{\underline{0}}]$	$[\underset{1}{y_P} : \underset{z}{\ell_{1,0} \cdot x_P} : \underset{z^2}{\underline{0}} : \underset{z^3}{\ell_{0,0}} : \underset{z^4}{\underline{0}} : \underset{z^5}{\underline{0}}]$
		T_3	$(x', y') \mapsto (wx', uwzy')$	$[\underset{1}{y_P} : \underset{w}{\underline{0}} : \underset{w^2}{\underline{0}} \mid \underset{1}{-\lambda \cdot u \cdot x_P} : \underset{w}{-\nu \cdot u} : \underset{w^2}{\underline{0}}]$	$[\underset{1}{y_P} : \underset{w}{\underline{0}} : \underset{w^2}{\underline{0}} \mid \underset{1}{\ell_{1,0} \cdot u \cdot x_P} : \underset{w}{\ell_{0,0} \cdot u} : \underset{w^2}{\underline{0}}]$

Table 5.2: Details of the Miller line function depending on the choice of tower (and twist constant).

"untwist" $Q' \in E'$ to $Q \in E$ for the line function computation, or choosing to put both points on the twist for the entire routine as in Theorem 3.1. The nature of the proposed tower actually means that there is a significant difference between the simplicity of the twisting and untwisting isomorphisms, and one option will be more desirable to implement than the other. For example, when using T_1 and $b' = -bv$, the untwisting isomorphism ψ is $\psi : (x', y') \mapsto (x/z^2, y/w) = (wvz(u-1)/2 \cdot x', wv(1-u)/2 \cdot y')$, which is more annoying to code (and theoretically slightly slower) than using $\psi^{-1} : (x, y) \mapsto (z^2x, wy)$ to twist the second argument P instead. On the other hand, if the twist is given as $E' : y^2 = x^3 \pm b/v$, as is the cases when $x_0 \equiv 7, 64 \bmod 72$, then it is the untwisting isomorphism that is clearly preferable.

5.3 Timings

This section provides timings of a plain C implementation of the (optimal) ate pairing on BLS curves with embedding degree $k = 24$ and parameter $x_0 \equiv$

16 mod 72. We give timings for field operations of base field, full extension field and all intermediate extension fields. Timings for pairings are split up into timing for the Miller loop, the final exponentiation, and the complete pairing. The number denoted "Product" refers to the time per pairing in a product of 20 pairings. This is intended to give some indication of the savings that are obtained when optimising the computation of a pairing product in a protocol that requires one (see [LMN10, §4.3] or [Sco11] for more details).

Timings are given for curves at security levels 192, 224 and 256 bits in Table 5.3, Table 5.4 and Table 5.5 respectively. Numbers in the names of curves give the bit size of r and that of q, i.e. the curve pfc-bls256-q319-k24 is a pairing-friendly BLS curve with a 256-bit group order r defined over a prime field of size 319 bits and embedding degree $k = 24$.

All measurements were done on an Intel Core 2 Duo CPU (E6850) running 64-bit Windows 7 at 3.00 GHz with 4GB RAM.

	add		**sub**		**M**		**S**		**I**	
	cyc	μs	cyc	μs	cyc	μs	cyc	μs	cyc	μs
$\mathbb{F}_p$	320	0.10	248	0.08	1112	0.34	1026	0.36	20320	6.83
$\mathbb{F}_{p^2}$	540	0.18	487	0.16	4627	1.52	4414	1.44	52334	17.16
$\mathbb{F}_{p^4}$	1068	0.35	942	0.32	21722	7.21	20532	6.76	131246	43.65
$\mathbb{F}_{p^{12}}$	3510	1.19	2750	0.93	163570	54.83	154325	51.21	744451	247.18
$\mathbb{F}_{p^{24}}$	6269	2.15	5536	1.85	539026	179.70	506029	168.13	2326049	777.80
	Pairings		**Miller loop**		**Final exp.**		**Single pairing**		**Product**	
	cyc		30,335,982		97,561,935		127,897,917		24,124,734	
	ms		10.00		32.62		42.62		8.05	

Table 5.3: Cycle counts and timings: pfc-bls384-q478-k24.

5.4 Summary of contributions

This was the first work to consider finer aspects of implementation for pairings at the 256-bit security level. Our method of identifying attractive subfamilies in this chapter is simple and effective, and sets the foundations for our more thorough approach in the following chapter.

Section 5.2 treats a subtle but consequential point of optimising the code for computations related to line functions depending on the nature of the twist. Scott's note [Sco09, §3] suggests that there is a probable preference between the

	add		**sub**		**M**		**S**		**I**	
	cyc	μs	cyc	μs	cyc	μs	cyc	μs	cyc	μs
$\mathbb{F}_p$	334	0.11	288	0.09	1235	0.41	1232	0.41	24918	8.23
$\mathbb{F}_{p^2}$	606	0.20	541	0.19	5529	1.82	5202	1.72	61942	20.70
$\mathbb{F}_{p^4}$	1219	0.41	1107	0.36	26159	8.64	24532	8.19	158822	52.81
$\mathbb{F}_{p^{12}}$	3710	1.36	3124	1.11	194825	65.07	183680	61.00	895588	297.69
$\mathbb{F}_{p^{24}}$	7601	2.53	6678	2.24	641578	215.21	603282	202.31	2801070	936.49
	Pairings		**Miller loop**		**Final exp.**		**Single pairing**		**Product**	
	cyc		42,945,862		157,289,781		200,235,643		35,992,064	
	ms		14.64		52.65		67.29		11.99	

Table 5.4: Cycle counts and timings: pfc-bls448-q559-k24a.

	add		**sub**		**M**		**S**		**I**	
	cyc	μs	cyc	μs	cyc	μs	cyc	μs	cyc	μs
$\mathbb{F}_p$	367	0.12	393	0.10	1475	0.52	1465	0.49	28511	9.70
$\mathbb{F}_{p^2}$	659	0.22	587	0.20	6272	2.09	5973	1.99	71990	23.96
$\mathbb{F}_{p^4}$	1272	0.42	1146	0.38	29253	9.70	27512	9.24	181534	60.34
$\mathbb{F}_{p^{12}}$	3884	1.36	3443	1.15	218255	72.04	209100	68.05	1010078	337.78
$\mathbb{F}_{p^{24}}$	7644	2.68	6982	2.32	708585	236.46	665836	221.82	3127211	1041.84
	Pairings		**Miller loop**		**Final exp.**		**Single pairing**		**Product**	
	cyc		53,827,736		168,824,048		222,651,784		41,951,965	
	ms		17.98		56.29		74.27		13.99	

Table 5.5: Cycle counts and timings: pfc-bls513-q639-k24a.

two twist types because of the difference in the simplicities of the isomorphisms. Specifically, he says that type D sextic twists ($y^2 = x^3 + b/v$ where $x^6 - v$ is the sextic extension) are likely to be preferable to type M twists ($y^2 = x^3 + bv$) because of the simplicity of the untwisting isomorphism in the former case. In contrast, we argue that there is no real preference, since the consequence of Theorem 3.1 is that we can either twist or untwist. Table 5.2 illustrates the simplicity of the twisting isomorphism in the case of type M twists, and the simplicity of the untwisting isomorphism in the case of type D twists. This same association occurs across the families considered in the next chapter (for quartic twists too), and should therefore be kept in mind when one is implementing a curve with a type M or type D twist (see Section 6.1.4).

Chapter 6

Particularly friendly members of family trees

As we discussed in Section 2.5, the last decade has witnessed many clever constructions of parameterised families of pairing-friendly elliptic curves that now enable implementors targeting a particular security level to gather suitable curves in bulk. However, as we saw in the previous chapter, choosing the best curves from a (usually very large) set of candidates belonging to any particular family involves juggling a number of efficiency issues, such as the nature of binomials used to construct extension fields, the Hamming weight of key pairing parameters and the existence of compact generators in the pairing groups. In light of these issues, Pereira *et al.* [PJNB11] considered the best subfamilies of $k = 12$ BN curves to especially target the 128-bit security level, whilst our slightly different approach in Chapter 5 considered the best subfamilies of $k = 24$ BLS curves to target 256-bit pairings. In this chapter we closely investigate the other eight attractive families with $8 \leq k < 50$, and systematically sub-divide each family into its *family tree*, branching off until concrete subfamilies are highlighted that simultaneously provide highly-efficient solutions to all of the above computational issues.

In the context of cryptography, the most efficient pairings make use of large prime order subgroups of elliptic curves $E/\mathbb{F}_q$. For optimal performance, pairings at different security levels demand elliptic curves with different embedding degrees (see Figure 2.39), so in their widely used taxonomy [FST10], Freeman,

Scott and Teske present the best constructions of pairing-friendly curves corresponding to all embedding degrees $1 \leq k \leq 50$. For current levels of security, and for those in the foreseeable future, the optimal curve choices come from *parameterised families* of ordinary (non-supersingular) curves over prime fields. This means that the field size and the number of points on the curve are parameterised as $q(x)$ and $n(x)$ respectively. If $n(x)$ is reducible then $n = n(x_0)$ will not be prime in general, so we usually also write down $r(x)$, the largest irreducible factor of $n(x)$. The straightforward way to find curves within a given family is to seek x_0's of appropriate size such that $q(x_0)$ and $r(x_0)$ are prime (or $r(x_0)$ is almost prime), at which point we have suitable pairing-friendly curves with $r(x_0) \mid n(x_0) = \#E(\mathbb{F}_{q(x_0)})$. If left for a few minutes, a simple code that does exactly this can return many pairing-friendly curves, and in most cases this is just a tiny fraction of the potential curves that could be used to target a particular security level. A natural problem that faces serious implementors then, is how to find and use only the very best curves within a family: this is the motivation for our work in this chapter.

Aside from the $k = 12$ BN and $k = 24$ BLS families that have already been considered, here we thoroughly treat the other eight stand-out candidates for pairing implementations with $8 \leq k < 50$. Since it is widely accepted that embedding degrees of the form $k = 2^i 3^j$ perform most efficiently [KM05] (see also [FST10, Table 6]), we look at the Kachisa-Schaefer-Scott (KSS) families [KSS08] with $k = 16$, $k = 18$, $k = 32$ and $k = 36$, and at the BLS families [BLS02] with $k = 27$ and $k = 48$. Following recent work by Aranha *et al.* [AFCK$^+$12] that targets 192-bit pairings, we also include the BLS family with $k = 12$. In addition, thanks to a suggestion made to us by Michael Scott, we also consider the Brezing-Weng family [BW05] with $k = 8$; a prime candidate for pairings at the (triple-DES equivalent) 112-bit security level. In all eight scenarios, our systematic approach allows us to point out several implementation-friendly subfamilies that simultaneously offer all of the desirable properties mentioned above, and many more (see [PJNB11, §1]). As a resource for implementors, we provide many examples of pairing-friendly curves according to our favourite *picks* from each tree, which are all readily found within the corresponding families. These are found in Appendix B.

We organise this chapter as follows. In Section 6.1 we begin by detailing how to read and use the family trees, as well as the main advantages of our approach.

The next eight sections (§6.2-§6.9) are dedicated to the eight selected families; in each of these sections we present the corresponding family tree and our favourite picks from it. In Section 6.10 we give our recommendations.

6.1 Family Trees

For all of the parameterised families considered in this work, the polynomials for the prime field characteristic $q(x)$ and/or the elliptic curve group order $n(x)$ have denominators, i.e. $q(x), n(x) \in \mathbb{Q}[x]$, but $q(x), n(x) \notin \mathbb{Z}[x]$. This means that only a subset of $x \in \mathbb{Z}$ will be such that $q(x)$ and $n(x)$ can both take on integers, and in all cases this subset is simply defined by some congruency condition, say $x \equiv a \bmod u$. In the simplest scenario, one then kick-starts a search for pairing-friendly curves by initialising an appropriately sized $x_0 \equiv a \bmod u$, and iterating with $x_0 \leftarrow x_0 + u$ until $q(x)$ is prime and $r(x)$, the largest irreducible factor of $n(x)$, is either a prime or almost prime. At this stage it is then possible to compute the curve equation, find simple irreducible polynomials over $\mathbb{F}_q$ to tower up to the full extension field, and determine which twisted curve is the correct one. In general, from one successful x_0 value (i.e. pairing-friendly curve) to the next, all of these parameters are likely to be different. In the end, there are many different combinations of the necessary pairing parameters to choose from, and therefore most of the curves encountered in a basic search will inevitably be discarded in favour of the very best ones. The ideal alternative is to be able to prescribe the desired properties in advance, and only search for curves that are guaranteed to exhibit all of them. This way, searches will avoid a great deal of unnecessary testing and, over any given time, have a better chance of finding supreme curves.

6.1.1 Branching out

The natural way to proceed towards this goal is to start by subdividing the major equivalence class $x \equiv a \bmod u$ into smaller subclasses $x \equiv \{a + iu\}_{0 \leq i < v} \bmod uv$, and to individually separate each of the resulting subclasses again, repeating the process with the goal of arriving at subclasses where the curves found within it share identical parameters. There are three traits of the curve we aim to synchronise: the extension field tower used to represent $\mathbb{F}_{q^k}$ over $\mathbb{F}_q$, the curve equation, and the *type* of twist (which only has two options – see below). Thus,

we leave the twist classification until the end, so that subdivisions or *branchings* depend only on the towering choice and on the curve equation; at each stage, the choice of v that inflates the modulus above will be dictated by one or the other, or sometimes both. We always take the choice that we believe was most obvious, but argue that the end result doesn't matter; overall it will take the same sized inflation (and probably number of intermediate subdivisions) of the original modulus to determine the specific subclasses that give identical pairing parameters.

6.1.2 Extension field towers

For each family, we present between two and six stand-out towering options for the construction of $\mathbb{F}_{q^k}$. Our towers are presented using two binomials: one used for the first extension from $\mathbb{F}_q$ to either $\mathbb{F}_{q^2}$ or $\mathbb{F}_{q^3}$, and the other for the remainder of the extension up to $\mathbb{F}_{q^k}$. More often than not, this produces more preferable towers (faster extension field arithmetic) than if only one binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^k}$ was used to define the tower. For example, for any of the k considered in this chapter, it is easy to show that $x^k \pm 1$ will never be irreducible[1] in $\mathbb{F}_q[x]$. However, choosing instead a different degree k irreducible binomial $x^k + s$ ($s \neq \pm 1$), means that the quadratic extension (if applicable) from $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ can no longer be constructed optimally as $\mathbb{F}_q[u]/(u^2+1)$. Alternatively, defining the tower with two binomials allows for $\mathbb{F}_{q^2} = \mathbb{F}_q[u]/(u^2+1)$, and $\mathbb{F}_{q^k}$ being constructed as, say $\mathbb{F}_{q^2}[v]/(v^{k/2} - (u+1))$, which is clearly preferable (cf. [AKL+11,PJNB11] – or see Chapter 5). The towers T_i are described in the sections corresponding to each family, are given in (our) preferential order, and are marked red in the trees.

6.1.3 Curve equations

All families under consideration either have j-invariant 0 or 1728, meaning that the elliptic curve equation is defined by one constant: b in $y^2 = x^3 + b$ for $j = 0$ or a in $y^2 = x^3 + ax$ for $j = 1728$. In both cases, we always take the correct curve whose constant has the smallest absolute value, so any multiplications by it (if at all) incur the minimal number of $\mathbb{F}_q$ additions. In all scenarios herein, this results in fewer than 10 distinct a or b values that rear their heads most commonly. The curve constants used to subdivide congruences are the subscripts of the a_i or b_i

[1] Refer back to Prop. 5.2 for the $3 \nmid k$ cases, whilst the $3 \mid k$ case is obvious.

values marked blue in the trees.

6.1.4 Type of twist

If the binomial used to extend from the twisted subfield $\mathbb{F}_{p^d}$ to the full extension field $\mathbb{F}_{p^k}$ is $x^{k/d} - i$ with $i \in \mathbb{F}_{p^d}$, then Scott [Sco09] shows that for the case of quartic twists on $y^2 = x^3 + ax$ and sextic twists on $y^2 = x^3 + b$, the correct twist is either type M (multiplication) which is given by $y^2 = x^3 + ax \cdot i$ and $y^2 = x^3 + b \cdot i$ respectively, or type D (division) which is given by $y^2 = x^3 + ax/i$ and $y^2 = x^3 + b/i$ respectively. The only other case is the cubic twist for $k = 27$ in Section 6.6, where we define the type M twist as $y^2 = x^3 + b \cdot i^2$ and the type D twist as $y^2 = x^3 + b/i^2$; this is to force quadratic reciprocity of the element in the twisted subfield. The type of twist corresponding to any given congruency is found immediately above (or in rare cases besides) the subclass; this is marked dark green in the trees. There is no great difference or preference between the two, if they are dealt with correctly (refer back to Section 5.4).

6.1.5 Fruits

If a subclass or subclasses of a family share all three traits, we call them *fruits* and they are labelled light green in the trees. Any equivalence classes found in the same fruit *bunch* share the same three parameters described in the previous three paragraphs; all three parameters can be easily seen by following the branches back up to the top of the tree. The most immediate bold number found above any bunch is the modulus corresponding to the equivalence classes in the bunch. Any branchings that don't (yet) produce consistent pairing parameters for its congruencies are marked grey, and are called *unripe*. In almost all cases pursuing further branching of the unripe fruits gives either undesirable pairing parameters, or congruency classes that are too scarce for our recommendation.

6.1.6 Our "picks"

After presenting a family tree, we pick our favourite subfamilies in it, and give them a star rating (up to 5). Our choice is mostly influenced by the towering option, since we believe this has the greatest effect on the pairing efficiency. For each of our favourite subfamilies, we searched for compact representations of generators in both the elliptic curve groups $\mathbb{G}_1$ and $\mathbb{G}_2$. In most families, our

favourite subfamilies either exhibit one or the other, or both. In many cases there are several suitable generators, so we have put some of the extra options in Appendix D. For subfamilies where generators in either group aren't given, it does not mean that one or more compact generators doesn't exist; it just means that our (somewhat basic) searches weren't able to find any. Moreover, for any one particular curve belonging to the subfamily, there's still a good chance that compact generators (that don't apply to the entire subfamily) can be found. Beside each of our picks, we give approximate frequencies of the corresponding subfamily across the entire family of curves. Most of these percentages were calculated from somewhere between 5,000 and 175,000 example curves from each family, and are essentially always as we would expect, given the corresponding restriction on the original congruency. In Appendix B, we account for a wide range of security levels and provide comprehensive lists of low Hamming weight curves belonging to 5-star subfamilies.

6.1.7 Advantages

The tree approach is exhaustive and complete, i.e. the branching technique described above doesn't lose track of any congruencies, which means that every curve belonging to a family under consideration fits somewhere in the family tree presented. Another advantage of presenting the entire tree, rather than just presenting specific subfamilies, is that many implementors will only want to simultaneously assure some proper subset of the properties we used to form the family trees. Thus, one can group together separate bunches of the tree that share this subset of desirable properties, and ignore the other property/s that caused them to branch away from one another. As an example, suppose one is using affine coordinates for a pairing implementation on a $k = 18$ KSS curve of the form $y^2 = x^3 + b$. The curve and pairing arithmetic will therefore be independent of b (cf. [LMN10]), so if the implementors are not necessitating consistent compact generators in $\mathbb{G}_1$ or $\mathbb{G}_2$, then all bunches with identical towers and twist types (but different curve equations) could be grouped in the same search, and use the same ($\mathbb{F}_p$-independent) pairing code.

6.1.8 Proofs

Since the proofs are tedious, repetitive, and follow the same script as the proofs we gave in full in Chapter 5, they have been moved into Appendix C. We provide proofs of the irreducibility criterions for the extension field towers, which rely on elementary number theory (quadratic and cubic reciprocity modulo q) that is essentially due to Gauss and Euler. We also make constant use of the same helpful theorem due to Benger and Scott [BS10] that we used in the last chapter. Since all the elliptic curves within have special CM discriminants, we don't need the (deeper) more general CM theory for the correct curve equations, but instead draw heavily on Algorithms 3.4 and 3.5 of [RS10], which are also "essentially due to Gauss". For every family, the proofs of the twist type (which is always one of two options – see [HSV06, Prop. 8]) follow the recipe in the proof of Proposition 5.5, so we omit them for space considerations. We do not prove any non-existence or negative results, e.g. that an extension field tower which would clearly be preferred does not (always) apply to this congruency, but the reader should rest assured that we tried all such options, and this is indeed the case.

6.1.9 Other parameters

Along with $q(x)$, $r(x)$ and $n(x)$, the description of the polynomial parameterisations for each family include the trace of Frobenius $t(x)$, the $\mathbb{G}_1$ cofactor $h(x) = n(x)/r(x)$, and $f(x)$, which comes from the CM norm equation $4q = t^2 - Df^2$. The polynomials $f(x)$ and $t(x)$ are commonly used in the proofs.

6.1.10 x or x′

The four KSS families all start with congruences of the form $x \equiv \pm au \bmod bu$. For the purpose of simplicity, we replace x by $x' = x/u$ and work instead with the simpler expression $x' \equiv a \bmod b$. We therefore remind those making use of the results within to inflate the x' congruence back to the congruence in x when searching for curves, or alternatively update the polynomials for $q(x)$ and $n(x)$ to $p'(x')$ and $n'(x')$, etc.

6.2 Brezing-Weng $k = 8$ curves

The polynomial parameterisations for the Brezing-Weng family with $k = 8$ are:

$$\begin{aligned} q(x) &= (81x^6 + 54x^5 + 45x^4 + 12x^3 + 13x^2 + 6x + 1)/4; \qquad f(x) = 3x + 1; \\ t(x) &= -9x^3 - 3x^2 - 2x; \qquad n(x) = (9x^2 - 6x + 5) \cdot (9x^4 + 12x^3 + 8x^2 + 4x + 1)/4; \\ h(x) &= 9x^2/2 - 3x + 5/2; \qquad r(x) = (9x^4 + 12x^3 + 8x^2 + 4x + 1)/2. \end{aligned} \tag{6.1}$$

We found three towers that were often applicable to the congruences found in the family tree. They are defined by the binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ and the binomial from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^8}$ (see Table 6.1). The polynomials for $q(x)$ and $n(x)$ in (6.1) insist

$\mathbb{F}_q \xrightarrow{\mathbb{F}_q[u]/(u^2+u_i)} \mathbb{F}_{q^2} \xrightarrow{\mathbb{F}_{q^2}[v]/(v^4-v_i)} \mathbb{F}_{q^8}$			
T_i	T_1	T_2	T_3
(u_i, v_i)	$(2, u)$	$(3, u)$	$(5, u)$

Table 6.1: Efficient towering options in the $k = 8$ Brezing-Weng tree.

that x is odd, so we begin with the congruence $x \equiv 1 \bmod 2$ and branch off into sub-congruences to form the family tree in Figure 6.1 (see Appendix C for the proofs). We pick several fruit bunches that offer particularly friendly parameters for the pairing computation, and provide compact generators in the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ where we found them (see Table 6.2). The frequencies in the final column were calculated from over 128,000 different Brezing-Weng curves and are entirely as expected. Our 5-star picks constitute approximately 50% of the entire family.

rating	equiv. class for x	tower	a	twist type	$\mathbb{G}_1$ gen. $[h](\cdot,\cdot)$	$\mathbb{G}_2'$ gen. $[h'](\cdot,\cdot)$	more §D.1	%
	1 mod 16	T_1	1	D	-	$(u, \sqrt{1-2u})$	(i)	12.5
	9 mod 16	T_1	1	M	-	$\left(u-1, \sqrt{3}\right)$	(ii)	12.6
⋆⋆⋆⋆⋆	3 mod 16	T_1	-2	D	$(1, \sqrt{-1})$	$\left(u-1, \sqrt{3}\right)$	(iii)	12.3
	11 mod 16	T_1	2	M	-	$(1, \sqrt{1+2u})$	(iv)	12.5
	7 mod 24	T_2	3	D	$(1, 2)$	$(1, 1+3/u)$	(v)	8.4
	13 mod 48	T_2	2	M	$\left(4, 6\sqrt{2}\right)$	$(u-1, \sqrt{2-2u})$	(vi)	4.1
	29 mod 48	T_2	2	D	$\left(4, 6\sqrt{2}\right)$	-	(vii)	4.2
⋆⋆⋆⋆	$\{47, ..., 239\}_4 \bmod 240$	T_2	6	M	-	$(1, \sqrt{1+6u})$		3.4
	$71, 85, 191 \bmod 240$	T_2	5	M	$\left(2, 2\sqrt{3}\right)$	-	$(viii)$	2.4
	5 mod 240	T_2	5	D	$\left(2, 2\sqrt{3}\right)$	$\left(5, \sqrt{125+25/u}\right)$	(ix)	0.9

Table 6.2: Our favourite picks from the $k = 8$ Brezing-Weng tree.

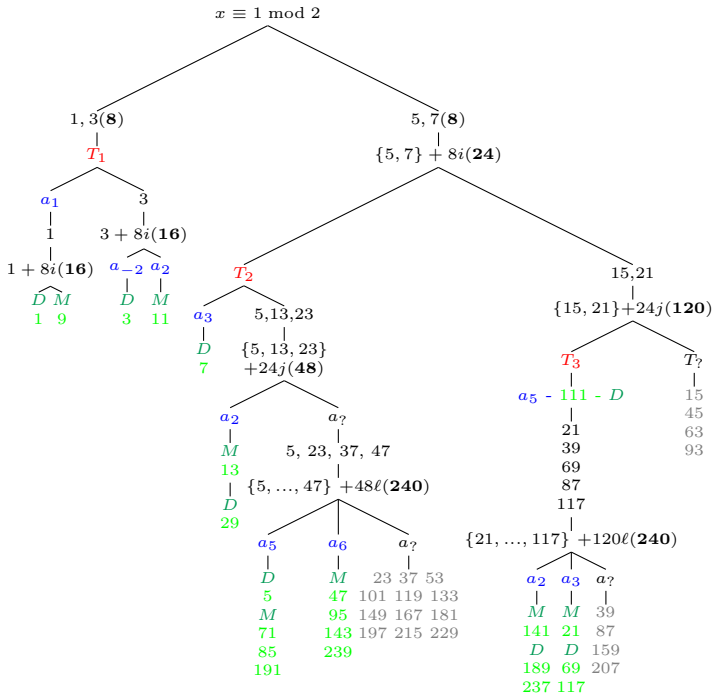


Figure 6.1: The $k = 8$ Brezing-Weng tree.

6.3 BLS $k = 12$ curves

The polynomial parameterisations for BLS family with $k = 12$ are:

$$q(x) = (x-1)^2(x^4 - x^2 + 1)/3 + x; \quad n(x) = (x-1)^2(x^4 - x^2 + 1)/3; \quad t(x) = x + 1;$$
$$h(x) = (x-1)^2/3; \qquad f(x) = (x-1)(2x^2 - 1)/3; \qquad r(x) = x^4 - x^2 + 1. \tag{6.2}$$

We found six towers that were often applicable to the congruences found in the

$\mathbb{F}_q \xrightarrow{\mathbb{F}_q[u]/(u^2+u_i)} \mathbb{F}_{q^2} \xrightarrow{\mathbb{F}_{q^2}[v]/(v^6-v_i)} \mathbb{F}_{q^{12}}$						
T_i	T_1	T_2	T_3	T_4	T_5	T_6
(u_i, v_i)	$(1, u+1)$	$(1, u+2)$	$(1, u+3)$	$(2, u)$	$(2, u+2)$	$(5, u)$

Table 6.3: Efficient towering options in the $k = 12$ BLS tree.

family tree. They are defined by the binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ and the binomial from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^{12}}$ (see Table 6.3). The polynomials for $q(x)$ and $n(x)$ in (6.2) insist that $x \equiv 1 \bmod 3$, so we begin with this congruence and branch off into sub-congruences to form the family tree in Figure 6.2 (see Appendix C for the proofs). To fit the tree in, one of the branches has been snapped off and is on its

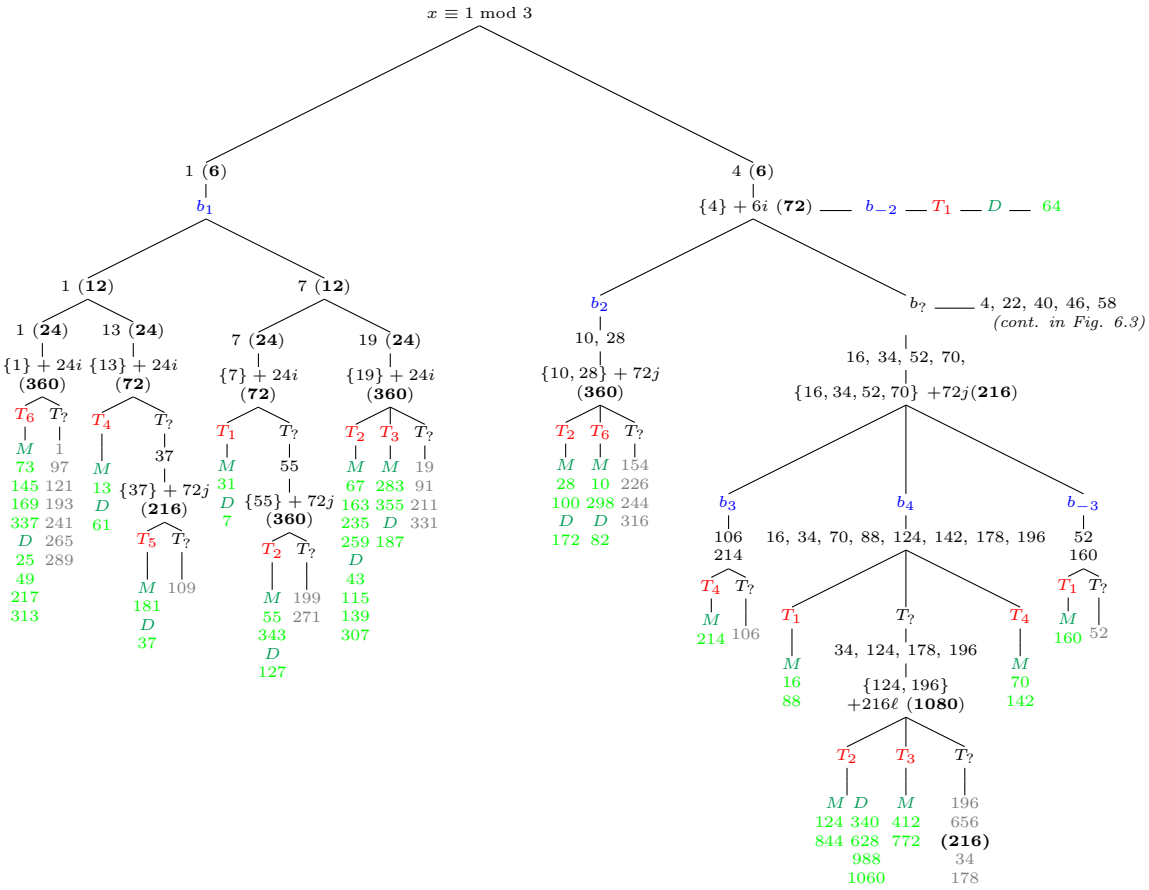


Figure 6.2: The $k = 12$ BLS tree.

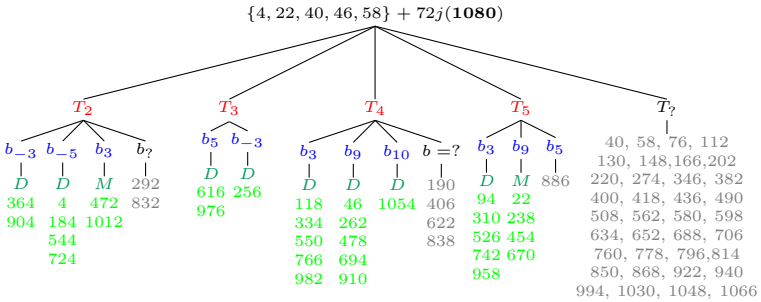


Figure 6.3: Another branch of the $k = 12$ BLS family tree.

own in Figure 6.3. We pick several fruit bunches that offer particularly friendly parameters for the pairing computation, and provide compact generators in the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ where we found them (see Table 6.4). The frequencies in the final column were calculated from over 170,000 different BLS curves and are entirely as expected. Our 5-star picks constitute approximately 17% of the entire family.

rating	equiv. class for x	tower	b	twist type	$\mathbb{G}_1$ gen. $[h](\cdot,\cdot)$	$\mathbb{G}_2'$ gen. $[h'](\cdot,\cdot)$	more §D.2	%
	$64 \bmod 72$	T_1	-2	D	$(-1, \sqrt{-3})$	$\left(1, \sqrt{\frac{u-1}{u+1}}\right)$		4.2
	$31 \bmod 72$	T_1	1	M	-	$(-1, \sqrt{u})$		4.1
⋆⋆⋆⋆⋆	$7 \bmod 72$	T_1	1	D	-	-		4.2
	$16, 88 \bmod 216$	T_1	4	M	-	$(-1, \sqrt{4u+3})$		2.8
	$160 \bmod 216$	T_1	-3	M	$(-1, \sqrt{-2})$	$(-1, \sqrt{-3u-4})$	(i)	1.4
	$\{67, ..., 259\}_4 \bmod 360$	T_2	1	M	$\left(1, \sqrt{2}\right)$	$(-1, \sqrt{u+1})$		3.3
	$55, 343 \bmod 360$	T_2	1	M	-	$(1, \sqrt{u+3})$		1.7
	$\{43, ..., 307\}_4 \bmod 360$	T_2	1	D	$\left(1, \sqrt{2}\right)$	$\left(1, \sqrt{\frac{u+3}{u+2}}\right)$		3.3
	$127 \bmod 360$	T_2	1	D	-	$\left(-1, \sqrt{-\frac{u+1}{u+2}}\right)$		0.8
⋆⋆⋆⋆	$28, 100 \bmod 360$	T_2	2	M	$(-1, 1)$	-		1.7
	$172 \bmod 360$	T_2	2	D	$(-1, 1)$	-		0.8
	$124, 844 \bmod 1080$	T_2	4	M	-	$(-2, 2\sqrt{u})$		0.6
	$\{340, ..., 1060\}_4 \bmod 1080$	T_2	4	D	-	$\left(-1, \sqrt{\frac{2-u}{2+u}}\right)$		1.1
	$283, 355 \bmod 360$	T_3	1	M	$\left(1, \sqrt{2}\right)$			1.7
⋆⋆⋆	$187 \bmod 360$	T_3	1	D	$\left(1, \sqrt{2}\right)$	$\left(-1, \sqrt{-\frac{u+2}{u+3}}\right)$		0.8

Table 6.4: Our favourite picks from the $k = 12$ BLS tree.

6.4 KSS $k = 16$ curves

The polynomial parameterisations for KSS family with $k = 16$ are:

$$\begin{aligned} q(x) &= (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980; \\ r(x) &= x^8 + 48x^4 + 625; \quad t(x) = (2x^5 + 41x + 35)/35; \quad h(x) = (x^2 + 2x + 5)/980; \\ n(x) &= (x^2 + 2x + 5)(x^8 + 48x^4 + 625)/980; \quad f(x) = (x^5 + 5x^4 + 38x + 120)/35. \end{aligned} \tag{6.3}$$

There were three common towers found in the family tree. They are defined by the binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ and the binomial from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^{16}}$ (see Table 6.5). The polynomials for $q(x)$ and $n(x)$ in (6.3) insist that $x \equiv \pm 25 \bmod 70$, so for

$\mathbb{F}_q \xrightarrow{\mathbb{F}_q[u]/(u^2+u_i)} \mathbb{F}_{q^2} \xrightarrow{\mathbb{F}_{q^2}[v]/(v^8-v_i)} \mathbb{F}_{q^{16}}$			
T_i	T_1	T_2	T_3
(u_i, v_i)	$(2, u)$	$(3, u)$	$(5, u)$

Table 6.5: Efficient towering options in the $k = 16$ KSS tree.

simplicity we rescale $x' = x/5$ and begin with $x' \equiv \pm 5 \bmod 14$, branching off into sub-congruences to form the family tree in Figure 6.4 (see Appendix C for the proofs). It is easy to see that $r(x)$ always has $2 \cdot 5^4 \cdot 7^2$ as a factor, so this division is necessary for (the updated) $r(x)$ to represent primes. We pick several fruit

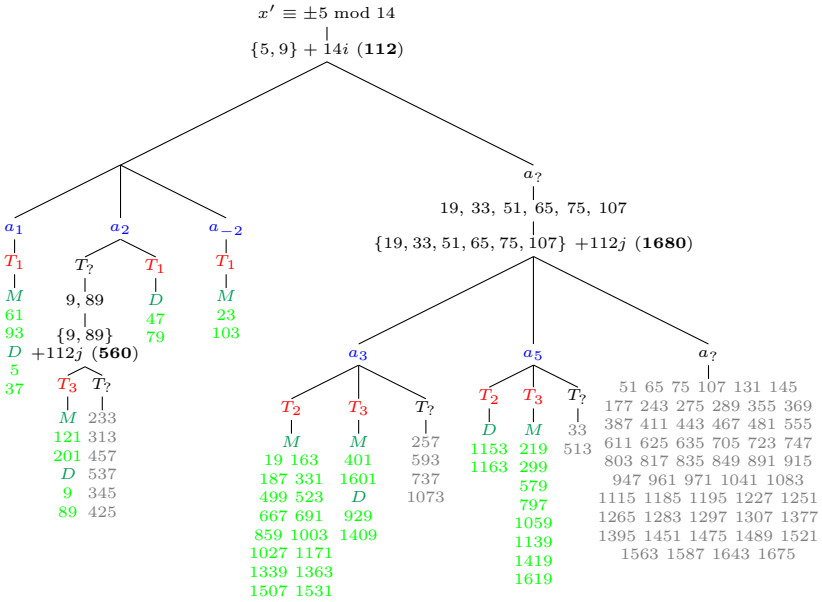


Figure 6.4: The $k = 16$ KSS family tree.

bunches that offer particularly friendly parameters for the pairing computation, and provide compact generators in the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ where we found them (see Table 6.6). The frequencies in the final column were calculated from over 13,000 different KSS curves with $k = 16$ and are entirely as expected. Our 5-star picks constitute approximately 50% of the entire family. The generators in $\mathbb{G}'_2 = E'(\mathbb{F}_{q^4})$ use $v \in \mathbb{F}_{q^4}$, where $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[v]/(v^2 - u)$.

rating	equiv. class for x' ($x' = x/5$)	tower	a	twist type	$\mathbb{G}_1$ gen. $[h](\cdot,\cdot)$	$\mathbb{G}'_2$ gen. $[h'](\cdot,\cdot)$	%
	$61, 93 \bmod 112$	T_1	1	M	-	$\left(v-1, \sqrt{(v-1)^3 + v(v-1)}\right)$	12.2
	$5, 37 \bmod 112$	T_1	1	D	-	$\left(-v, \sqrt{-v^3 - 1}\right)$	12.7
⋆⋆⋆⋆⋆	$47, 79 \bmod 112$	T_1	2	D	-	$\left(2/v, \sqrt{\frac{8}{v^3} + \frac{4}{v^2}}\right)$	12.1
	$23, 103 \bmod 112$	T_1	-2	M	$(1, \sqrt{-1})$	-	13.1
⋆⋆⋆⋆	$\{19, ..., 1531\}_{16} \bmod 1680$	T_2	3	M	$(1, 2)$	$\left(3/v, \sqrt{\frac{27}{v^3} + \frac{9}{v^2}}\right)$	7.9
⋆⋆⋆	$1153, 1633 \bmod 1680$	T_2	5	D	$\left(2, 2\sqrt{3}\right)$	-	0.9

Table 6.6: Our favourite picks from the $k = 16$ KSS tree.

6.5 KSS $k = 18$ curves

The polynomial parameterisations for KSS family with $k = 18$ are:

$$q(x) = (x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401)/21; \quad (6.4)$$
$$r(x) = x^6 + 37x^3 + 343; \qquad t(x) = (x^4 + 16x + 7)/7; \qquad h(x) = (x^2 + 5x + 7)/21;$$
$$n(x) = (x^2 + 5x + 7)(x^6 + 37x^3 + 343)/21; \qquad f(x) = (5x^4 + 14x^3 + 94x + 259)/21.$$

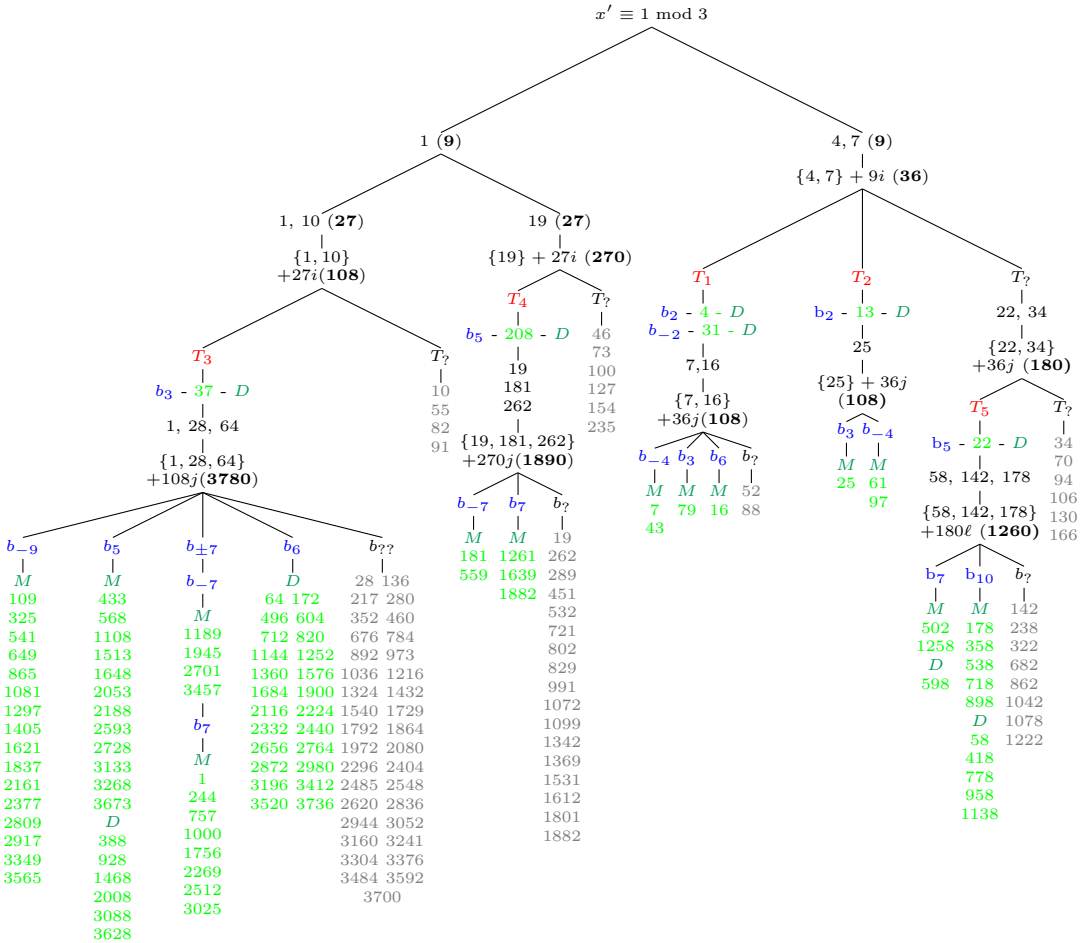


Figure 6.5: The $k = 18$ KSS family tree.

There were five common towers found in the family tree. They are defined by the cubic binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^3}$ and the binomial from $\mathbb{F}_{q^3}$ to $\mathbb{F}_{q^{18}}$ (see Table 6.7). The polynomials for $q(x)$ and $n(x)$ in (6.4) insist that $x \equiv 14 \bmod 42$, so we rescale $x' = x/14$ and begin with $x' \equiv \pm 1 \bmod 3$, branching off into sub-congruences to form the family tree in Figure 6.5 (see Appendix C for the proofs).

$\mathbb{F}_q \xrightarrow{\mathbb{F}_q[u]/(u^3+u_i)} \mathbb{F}_{q^3} \xrightarrow{\mathbb{F}_{q^3}[v]/(v^6-v_i)} \mathbb{F}_{q^{18}}$					
T_i	T_1	T_2	T_3	T_4	T_5
(u_i, v_i)	$(2, u)$	$(2, 2u)$	$(3, 2u)$	$(5, u)$	$(2, 5u)$

Table 6.7: Efficient towering options in the $k = 18$ KSS tree.

As it stands, $r(x)$ will always contain 7^3 as a factor, so this division is necessary for (the updated) $r(x)$ to represent primes.

Our favourite picks and the associated generators are in Table 6.8. The frequencies in the final column were calculated from over $25,000$ different KSS curves with $k = 18$ and are as expected. Our 5-star picks constitute approximately 27% of the entire family.

rating	equiv. class for $x' = x/14$	tower	b	twist type	$\mathbb{G}_1$ gen. $[h](\cdot,\cdot)$	$\mathbb{G}_2'$ gen. $[h'](\cdot,\cdot)$	more §D.3	%
⋆⋆⋆⋆⋆	4 mod 36	T_1	2	D	$(-1, 1)$	$\left(1, \sqrt{\frac{u+2}{u}}\right)$	(i)	8.4
	31 mod 36	T_1	-2	D	$(3, 5)$	$\left(1-u, \sqrt{(1-u)^3 - 2/u}\right)$	(ii)	7.9
	$7, 43$ mod 108	T_1	-4	M	$(2, 2)$	$\left(-2, 2\sqrt{-(u+2)}\right)$	(iii)	5.3
	79 mod 108	T_1	3	M	$(1, 2)$	$\left(-1, \sqrt{-1+3u}\right)$	(iv)	2.9
	16 mod 108	T_1	6	M	$\left(-1, \sqrt{5}\right)$	$\left(2, \sqrt{2+6u}\right)$		2.9
⋆⋆⋆⋆	13 mod 36	T_2	2	D	$(-1, 1)$	-		8.3
	$61, 97$ mod 108	T_2	-4	M	$(2, 2)$	$\left(2, 2\sqrt{2-2u}\right)$	(v)	5.5
	25 mod 108	T_2	3	M	$(1, 2)$	-		2.8
⋆⋆⋆	37 mod 108	T_3	3	D	$(1, 2)$	$\left(1, \sqrt{\frac{2u+3}{2u}}\right)$		2.9
	$\{109, ..., 3565\}_{16}$ mod 3708	T_3	-9	M	$\left(1, 2\sqrt{-2}\right)$	$\left(-3, 6\sqrt{-1}\right)$		1.4
	$\{568, ..., 3673\}_{13}$ mod 3708	T_3	5	M	$(-1, 2)$	-	(vi)	1.0
	$\{328, ..., 3628\}_6$ mod 3708	T_3	5	D	$(-1, 2)$	-	(vi)	0.5
	$\{1189, ..., 3457\}_4$ mod 3708	T_3	-7	M	$(2, 1)$	-	(vii)	0.4
	$\{1, ..., 2512\}_8$ mod 3708	T_3	7	M	$\left(7, 5\sqrt{14}\right)$	-		0.7
	$\{64, ..., 3736\}_{24}$ mod 3708	T_3	6	D	-	$\left(u-1, \sqrt{(u-1)^3 + 3/u}\right)$		1.9

Table 6.8: Our favourite picks from the $k = 18$ KSS tree.

6.6 BLS $k = 27$ curves

The polynomial parameterisations for BLS family with $k = 27$ are:

$$\begin{aligned} q(x) &= (x+1)^2(x^{18} - x^9 + 1)/3 - x^{19}; \quad r(x) = x^{18} - x^9 + 1; \\ f(x) &= (x^{10} - 2x^9 + x + 1)/3; \quad n(x) = (x^2 - x + 1)(x^{18} - x^9 + 1)/3; \\ t(x) &= -x^{10} + x + 1; \quad h(x) = (x^2 - x + 1)/3. \end{aligned} \tag{6.5}$$

There were three common towers found in the family tree. They are defined by the cubic binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^3}$ and the binomial from $\mathbb{F}_{q^3}$ to $\mathbb{F}_{q^{27}}$ (see Table 6.9). The polynomials for $q(x)$ and $n(x)$ in (6.5) insist that $x \equiv 1 \bmod 3$, which

$\mathbb{F}_q \xrightarrow{\mathbb{F}_q[u]/(u^3+u_i)} \mathbb{F}_{q^3} \xrightarrow{\mathbb{F}_{q^3}[v]/(v^9-v_i)} \mathbb{F}_{q^{27}}$			
T_i	T_1	T_2	T_3
(u_i, v_i)	$(3, u)$	$(5, u)$	$(7, u)$

Table 6.9: Efficient towering options in the $k = 27$ BLS tree.

we use to branch off into sub-congruences, forming the family tree in Figure 6.6 (see Appendix C for the proofs). As it stands, $r(x)$ will always contain 3 as a factor, so division by 3 is necessary for (the updated) $r(x)$ to represent primes. Our favourite picks and the associated generators are in Table 6.10.

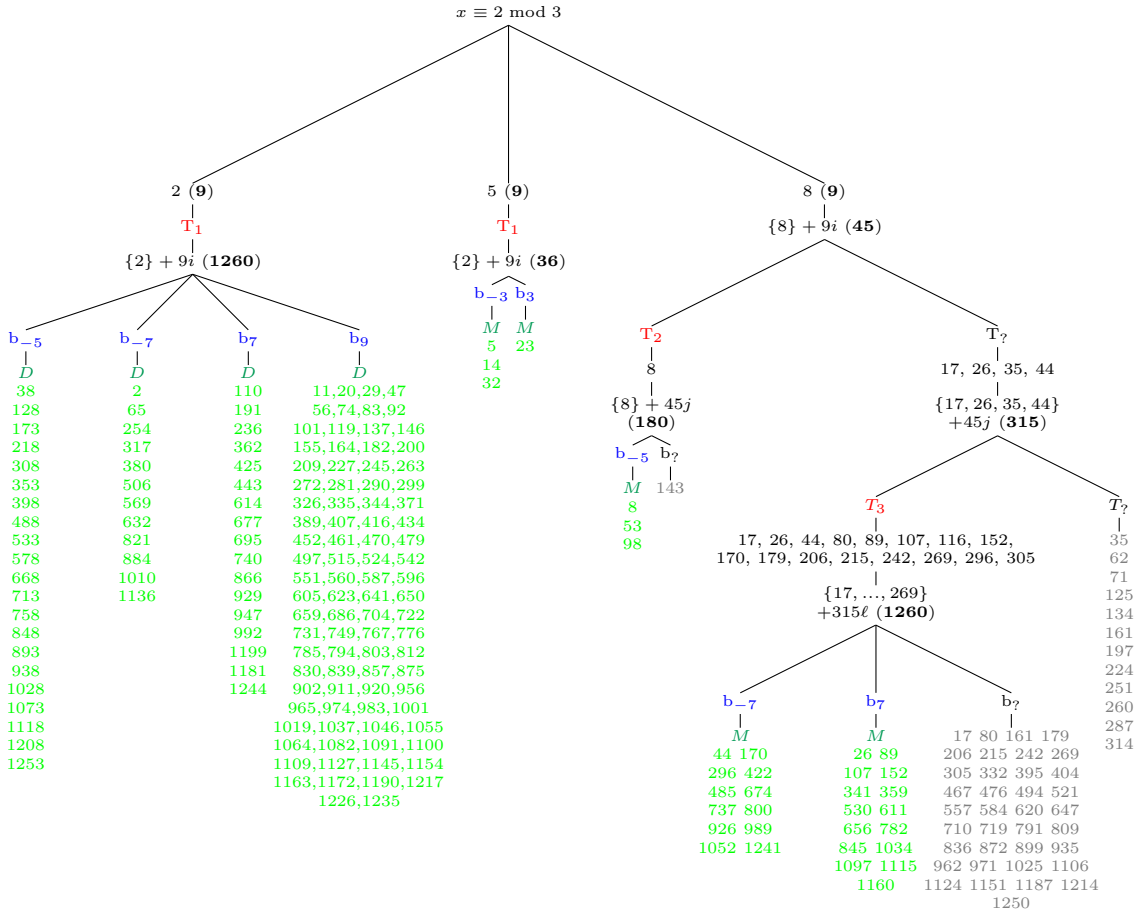


Figure 6.6: The $k = 27$ BLS family tree.

The frequencies in the final column were calculated from over $6,000$ different

BLS curves with $k = 27$ and behave as expected. Our 5-star picks constitute approximately 66% of the entire family. The generators in $\mathbb{G}_2' = E'(\mathbb{F}_{q^9})$ use $v \in \mathbb{F}_{q^9}$, where $\mathbb{F}_{q^9} = \mathbb{F}_{q^3}[v]/(v^3 - u)$.

rating	equiv. class for x	tower	b	twist type	$\mathbb{G}_1$ gen.	$\mathbb{G}_2'$ gen.	more §D.4	%
	5 mod 36	T_1	-3	M	-	-		8.5
	$14, 32$ mod 36	T_1	-3	M	$[h]\,(1, \sqrt{-2})$	-		16.8
	23 mod 36	T_1	3	M	$[h]\,(1, 2)$	-		7.8
⋆⋆⋆⋆⋆	$\{38, ..., 1253\}_{21}$ mod 1260	T_1	-5	D	-	-		5.0
	$\{2, .., 1136\}_{12}$ mod 1260	T_1	-7	D	$[h]\,(2, 1)$	-	(i)	3.1
	$\{110, .., 1244\}_{17}$ mod 1260	T_1	7	D	$[h]\,(-7, 4\sqrt{-21})$	-	(ii)	4.0
	$\{11, .., 1235\}_{89}$ mod 1260	T_1	9	D	$[h]\,(-2, 1)$	-	(iii)	21.0
⋆⋆⋆⋆	$8, 98$ mod 180	T_2	-5	M	$[h]\,(-3, 4\sqrt{-2})$	-		3.3
	53 mod 180	T_2	-5	M	-	-		1.67

Table 6.10: Our favourite picks from the $k = 27$ BLS tree.

6.7 KSS $k = 32$ curves

The polynomial parameterisations for the KSS family with $k = 32$ are:

$$\begin{aligned}
q(x) &= (x^{18} - 6x^{17} + 13x^{16} + 57120x^{10} - 344632x^9 + 742560x^8 + 815730721x^2 \\
&\qquad - 4948305594x + 10604499373)/2970292; \\
r(x) &= x^{16} + 57120x^8 + 815730721; \quad f(x) = 3x^9 - 13x^8 + 86158x - 371280; \\
n(x) &= (x^2 - 6x + 13)(x^{16} + 57120x^8 + 815730721)/2970292; \\
h(x) &= (x^2 - 6x + 13)/2970292. \quad t(x) = (-2x^9 - 56403x + 3107)/3107; \qquad (6.6)
\end{aligned}$$

There were only two common (and efficient) towers found in the family tree. They are defined by the quadratic binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ and the binomial from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^{32}}$ (see Table 6.11). The polynomials for $q(x)$ and $n(x)$ in (6.6)

$\mathbb{F}_q \xrightarrow{\mathbb{F}_q[u]/(u^2+u_i)} \mathbb{F}_{q^2} \xrightarrow{\mathbb{F}_{q^2}[v]/(v^{16}-v_i)} \mathbb{F}_{q^{32}}$		
T_i	T_1	T_2
(u_i, v_i)	$(2, u)$	$(3, u)$

Table 6.11: Efficient towering options in the $k = 32$ KSS tree.

insist that $x \equiv \pm 325 \bmod 6214$, so we rescale with $x' = x/13$ and begin with $x' \equiv \pm 25 \bmod 478$, which branches off into sub-congruences forming the family tree in

Figure 6.7 (see Appendix C for the proofs). As it stands, $r(x)$ will always contain $2 \cdot 13^8 \cdot 239^2$ as a factor, so we must divide this factor out before (the updated) $r(x)$ can represent primes. Our favourite picks and the associated generators are

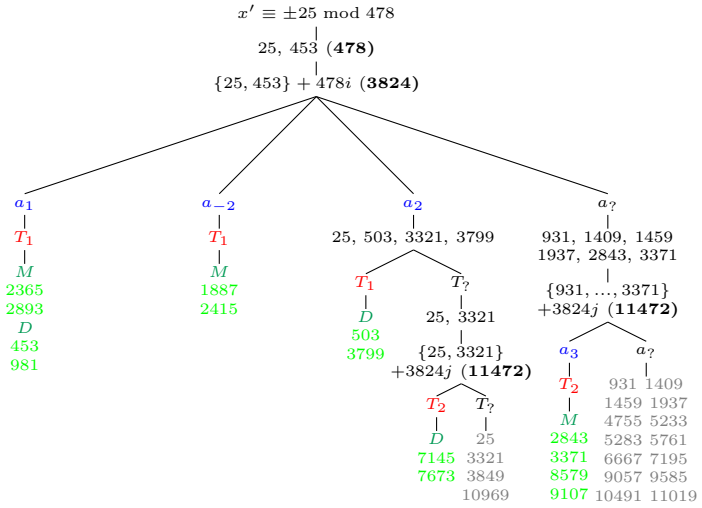


Figure 6.7: The $k = 32$ KSS family tree.

in Table 6.12. The frequencies in the final column were calculated from over 2,600 different KSS curves with $k = 32$ and are roughly as expected. Our 5-star picks constitute approximately 51% of the entire family. The generators in $\mathbb{G}_2' = E'(\mathbb{F}_{q^8})$ use $w \in \mathbb{F}_{q^8}$, where $\mathbb{F}_{q^8} = \mathbb{F}_{q^4}[w]/(w^2 - v)$, and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[v]/(v^2 - u)$.

rating	equiv. class for $x' = x/13$	tower	a	twist type	$\mathbb{G}_1$ gen. $[h](\cdot,\cdot)$	$\mathbb{G}_2'$ gen. $[h'](\cdot,\cdot)$	more §D.5	%
	$2365, 2893 \bmod 3824$	T_1	1	M	-	$\left(w-1, \sqrt{w^3 - 2w^2 + 2w - 1}\right)$		13.2
⋆⋆⋆⋆⋆	$453, 981 \bmod 3824$	T_1	1	D		$\left(w, \sqrt{w^3 + 1}\right)$		11.5
	$1887, 2415 \bmod 3824$	T_1	-2	M	$(-1, 1)$	-	(i)	13.3
	$503, 3799 \bmod 3824$	T_1	2	D	-	-		12.7
	$7145, 7673 \bmod 11472$	T_2	2	D	$\left(4, 6\sqrt{2}\right)$	-	(ii)	5.4
⋆⋆⋆⋆	$\{2843, \ldots, 9107\}_4$ $\bmod 11472$	T_2	3	M	$(1, 2)$	-	(iii)	13.0

Table 6.12: Our favourite picks from the $k = 32$ KSS tree.

6.8 KSS $k = 36$ curves

The polynomial parameterisations for the KSS family with $k = 36$ are:

$$\begin{aligned} q(x) &= (x^{14} - 4x^{13} + 7x^{12} + 683x^8 - 2510x^7 + 4781x^6 + 117649x^2 - 386569x \\ &+ 823543)/28749; \qquad r(x) = x^{12} + 683x^6 + 117649; t(x) = (259 + 757x + 2x^7)/259; \\ n(x) &= (x^2 - 4x + 7)(x^{12} + 683x^6 + 117649)/28749 \quad h(x) = (x^2 - 4x + 7)/28749; \\ f(x) &= (4x^7 - 14x^6 + 1255x - 4781)/777. \end{aligned} \tag{6.7}$$

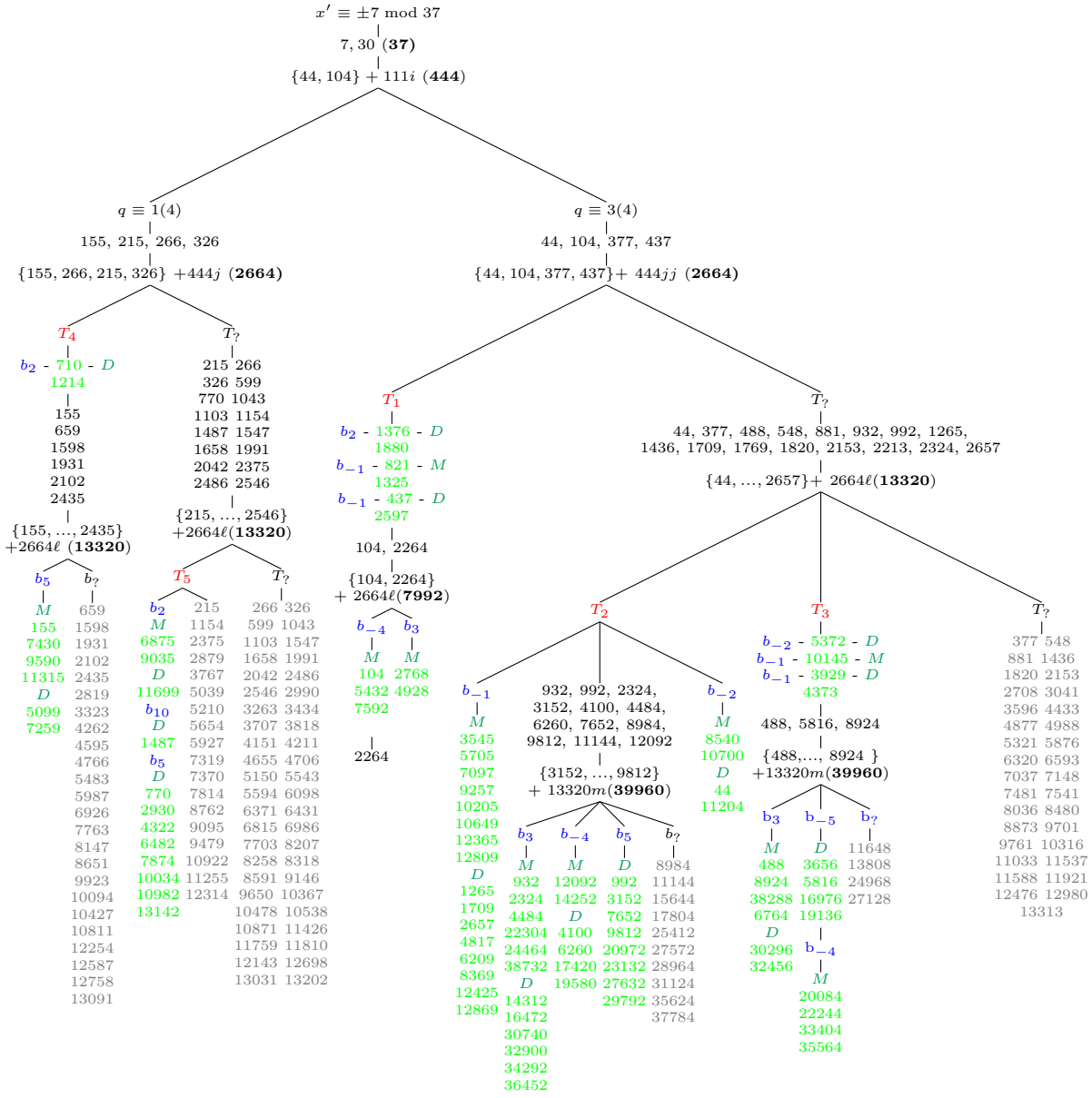


Figure 6.8: The $k = 36$ KSS family tree.

There are five common towers found in the family tree. They are defined by the quadratic binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ and the binomial from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^{36}}$ (see Table 6.13).

$\mathbb{F}_q \xrightarrow{\mathbb{F}_q[u]/(u^2+u_i)} \mathbb{F}_{q^2} \xrightarrow{\mathbb{F}_{q^2}[v]/(v^{18}-v_i)} \mathbb{F}_{q^{36}}$					
T_i	T_1	T_2	T_3	T_4	T_5
(u_i, v_i)	$(1, u+1)$	$(1, u+2)$	$(1, u+3)$	$(2, u)$	$(5, u)$

Table 6.13: Efficient towering options in the $k = 36$ KSS tree.

rating	equiv. class for $x' = x/7$	tower	b	twist type	$\mathbb{G}_1$ gen. $[h](\cdot,\cdot)$	$\mathbb{G}_2'$ gen. $[h'](\cdot,\cdot)$	§D.6	%
	$1376, 1880 \bmod 2664$	T_1	2	D	$(-1, 1)$	$\left(-1, \sqrt{\frac{2-v}{v}}\right)$		4.1
	$821, 1325 \bmod 2664$	T_1	-1	M	$\left(-1, \sqrt{-2}\right)$	$\left(1, \sqrt{1-v}\right)$		4.1
⋆⋆⋆⋆⋆	$437, 2597 \bmod 2664$	T_1	-1	D	$\left(-1, \sqrt{-2}\right)$	-		4.3
	$\{104, ..., 7592\}_4 \bmod 7992$	T_1	-4	M	$(2, 2)$	$\left(1-v, \sqrt{(1-v)^3 - 4v}\right)$	(i)	3.0
	$2768, 4928 \bmod 7992$	T_1	3	M	$(1, 2)$	$\left(1-v, \sqrt{(1-v)^3 + 3v}\right)$		1.1
	$\{3545, ..., 12809\}_8 \bmod 13320$	T_2	-1	M	-	$\left(v, \sqrt{u+2-v}\right)$		2.2
	$\{1265, ..., 12869\}_8 \bmod 13320$	T_2	-1	D	-	$\left(1/v, \sqrt{\frac{1}{u+2} - v}\right)$		3.4
	$8540, 10700 \bmod 13320$	T_2	-2	M	$(3, 5)$	$\left(1, \sqrt{1-2/v}\right)$	(ii)	1.0
	$44, 11204 \bmod 13320$	T_2	-2	D	$(3, 5)$	-	(ii)	0.9
⋆⋆⋆⋆	$\{932, ..., 38732\}_5 \bmod 39960$	T_2	3	M	$(1, 2)$	-	(iii)	0.7
	$\{14312, ..., 36452\}_6 \bmod 39960$	T_2	3	D	$(1, 2)$	-	(iii)	1.0
	$12092, 14252 \bmod 39960$	T_2	-4	M	$(2, 2)$	-	-	0.3
	$\{4100, ..., 19580\}_4 \bmod 39960$	T_2	-4	D	$(2, 2)$	-	-	0.6
	$\{992, ..., 29792\}_8 \bmod 39960$	T_2	5	D	-	$\left(-v, \sqrt{\frac{5-v^4}{v}}\right)$		1.1
	$5372 \bmod 13320$	T_3	-2	D	$(3, 5)$	-		0.5
	$10145 \bmod 13320$	T_3	-1	D	-	-		0.5
	$3929, 4373 \bmod 13320$	T_3	-1	M	-	-		0.8
⋆⋆⋆	$\{488, ..., 38288\}_4 \bmod 39960$	T_3	3	M	$(1, 2)$	-	-	0.5
	$30296, 32456 \bmod 39960$	T_3	3	D	$(1, 2)$	-	-	0.3
	$\{3656, ..., 19136\}_4 \bmod 39960$	T_3	-5	D	$\left(-3, 4\sqrt{-2}\right)$	-	(v)	0.6
	$\{20084, ..., 35564\}_4 \bmod 39960$	T_3	-4	M	$(2, 2)$	-	(vi)	0.6

Table 6.14: Our favourite picks from the $k = 36$ KSS tree.

The polynomials for $q(x)$ and $n(x)$ in (6.7) insist that $x \equiv \pm 49 \bmod 259$, so we rescale with $x' = x/7$ and begin with $x' \equiv \pm 7 \bmod 37$, which branches off into sub-congruences forming the family tree in Figure 6.8 (see Appendix C for the proofs). As it stands, $r(x)$ will always contain $7^6 \cdot 37^2$ as a factor, so we must divide this factor out before (the updated) $r(x)$ can represent primes. Our favourite picks and the associated generators are in Table 6.14. The frequencies in the final column were calculated from almost $7,000$ different KSS curves with $k = 36$ and are roughly as expected. Our 5-star picks constitute approximately

17% of the entire family. The generators in $\mathbb{G}_2' = E'(\mathbb{F}_{q^6})$ use $v \in \mathbb{F}_{q^6}$, where $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[v]/(v^3 - u)$.

6.9 BLS $k = 48$ curves

The polynomial parameterisations for the BLS family with $k = 48$ are:

$$q(x) = (x-1)^2(x^{16} - x^8 + 1)/3 + x; \quad r(x) = x^{16} - x^8 + 1; \quad t(x) = x + 1; \qquad (6.8)$$
$$n(x) = (x-1)^2(x^{16} - x^8 + 1)/3; \quad f(x) = (x-1)(2x^8 - 1)/3; \quad h(x) = (x-1)^2/3.$$

There are five common towers found in the family tree. They are defined by the quadratic binomial from $\mathbb{F}_q$ to $\mathbb{F}_{q^2}$ and the binomial from $\mathbb{F}_{q^2}$ to $\mathbb{F}_{q^{48}}$ (see Table 6.15). The polynomials for $q(x)$ and $n(x)$ in (6.8) insist that $x \equiv \pm 1 \bmod 3$,

$\mathbb{F}_q \xrightarrow{\mathbb{F}_q[u]/(u^2+u_i)} \mathbb{F}_{q^2} \xrightarrow{\mathbb{F}_{q^2}[v]/(v^{24}-v_i)} \mathbb{F}_{q^{48}}$					
T_i	T_1	T_2	T_3	T_4	T_5
(u_i, v_i)	$(1, u+1)$	$(1, u+2)$	$(2, u)$	$(2, u+2)$	$(5, u)$

Table 6.15: Efficient towering options in the $k = 48$ BLS tree.

which branches off into sub-congruences forming the family tree in Figure 6.9 (see Appendix C for the proofs). Our favourite picks and the associated generators

rating	equiv. class for x	tower	b	twist type	$\mathbb{G}_1$ gen. $[h](\cdot,\cdot)$	$\mathbb{G}_2'$ gen. $[h'](\cdot,\cdot)$	more §D.7	% %
	64 mod 72	T_1	-2	D	$(3, 5)$	$\left(1-2/w, \sqrt{(1-2/w)^3 - 2}\right)$	(i)	4.5
⋆⋆⋆⋆⋆	31 mod 72	T_1	1	M	-	$\left(w+1, \sqrt{(w+1)^3 + 1}\right)$	(ii)	4.5
	7 mod 72	T_1	1	D	-	-		4.4
	$55, 235, 259 \bmod 360$	T_2	1	M	-	-		2.0
	$115, 139 \bmod 360$	T_2	1	D	$\left(1, \sqrt{2}\right)$	-		1.4
	$100 \bmod 360$	T_2	2	M	-	-		0.7
	$\{4, ..., 724\}_4 \bmod 1080$	T_2	-5	D	$\left(-15, 26\sqrt{-5}\right)$	-		1.1
⋆⋆⋆⋆	$364, 700, 904 \bmod 1080$	T_2	-3	D	-	-		0.6
	$484 \bmod 1080$	T_2	-3	M	-	-		
	$124, 844 \bmod 1080$	T_2	4	M	-	$\left(2, \sqrt{8+4w}\right)$	(iii)	0.2
	$340, 1060 \bmod 1080$	T_2	4	D	-	$\left(1, \sqrt{1+4w}\right)$	(iv)	0.4
	$13 \bmod 72$	T_3	1	M	–	$\left(1, \sqrt{w+1}\right)$	(v)	3.9
	$61 \bmod 72$	T_3	1	D	–	-		4.1
⋆⋆⋆	$34, 178 \bmod 216$	T_3	4	M	$\left(-1, \sqrt{3}\right)$	-	(vi)	2.9
	$106 \bmod 216$	T_3	3	M	$(1, 2)$	-		1.4
	$10 \bmod 216$	T_3	3	D	$(1, 2)$	-		1.5

Table 6.16: Our favourite picks from the $k = 48$ BLS tree.

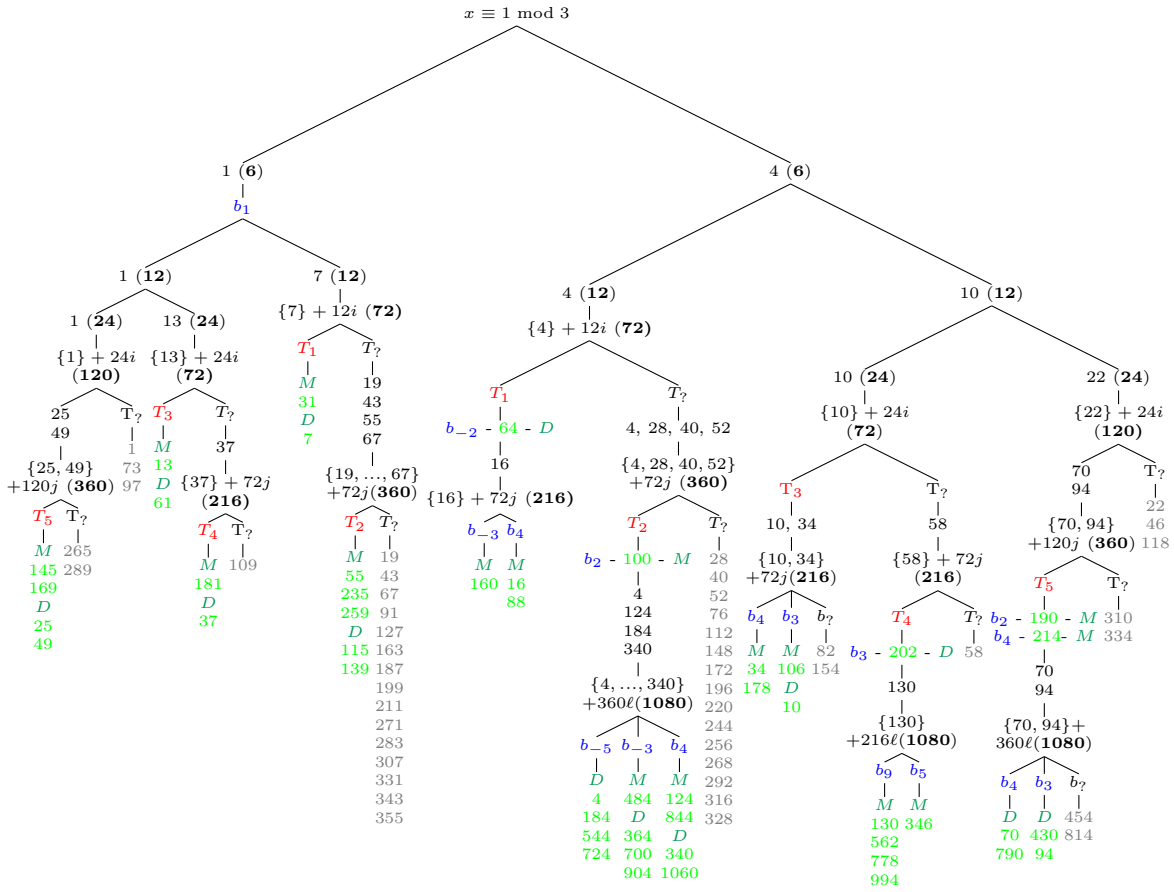


Figure 6.9: The $k = 48$ BLS family tree.

are in Table 6.16. The frequencies in the final column were calculated from over $5,000$ different BLS curves with $k = 48$ and are as expected. Our 5-star picks constitute approximately 51% of the entire family. The generators in $\mathbb{G}'_2 = E'(\mathbb{F}_{q^8})$ use $w \in \mathbb{F}_{q^8}$, where $\mathbb{F}_{q^8} = \mathbb{F}_{q^4}[w]/(w^2 - v)$, and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[v]/(v^2 - u)$.

6.10 Recommendations

For all curve families under consideration, as well as BN $k = 12$ and BLS $k = 24$ curves, Table 6.17 gives the approximate security level at which the DLP and ECDLP complexities are balanced. The ECDLP security is computed as half the bit-length of the group order r, whilst the calculation of the security in $\mathbb{F}_{q^k}$ comes directly from the formula in [Sma10, §6.2.1]. This gives a rough indication of which security level(s) a family is particularly suitable for, and where the family will best compete against other families. For such security levels, we point to

where examples of strong curves with low Hamming weights and implementation-friendly parameters can be found.

family	ρ-value ($\log q/\log r$)	x_0 (bits)	where security balance occurs: $E[r(x_0)]$ sec. (bits)	where security balance occurs: $\mathbb{F}_{q(x_0)^k}$ sec. (bits)	example curves: security levels (bits)	example curves: where to find curves
BW $k = 8$	1.5	60	121	121	112	Table B.1
BN $k = 12$	1	60	122	122	80 - 192	[PJNB11, §4]
BLS $k = 12$	1.5	85	170	170	192, 224	Table B.2, Table B.3
KSS $k = 16$	1.25	49	188	189	192, 224	Table B.2, Table B.3
KSS $k = 18$	1.33	74	217	217	192, 224	Table B.2, Table B.3
BLS $k = 24$	1.25	62	252	253	192 - 320	Appendix A
BLS $k = 27$	1.11	28	251	253	256, 288	Table B.4, Table B.5
KSS $k = 32$	1.125	41	304	302	288, 320	Table B.5, Table B.6
KSS $k = 36$	1.167	57	328	330	320, 352	Table B.6, Table B.7
BLS $k = 48$	1.125	49	392	390	352, 384	Table B.7, Table B.8

Table 6.17: Balancing ECDLP and DLP security in families, and where example curves are found.

Our advice on how to proceed agrees almost entirely with that at the end of Scott's note [Sco09, §6], who recommends first finding the optimal degree k binomial $x^k - i \in \mathbb{F}_q[x]$ to define the entire tower, before going searching for curves (that support this tower). The only difference in our recommendation is in the slight performance gain that's achieved when defining the tower using two binomials. Thus, we recommend choosing one or more of the subfamilies that guarantee our favourite tower choices (or yours), and restricting the search parameter x_0 to the corresponding congruences and to be of low Hamming weight before kick-starting a search. If, in addition, there is a preference in the size of a curve constant, the nature of the twist, or the existence of compact generators, then the subfamilies herein give a concrete way to also simultaneously prescribe these desired properties in advance.

Alternatively, the lists of curves in Appendix B should stand implementors in good stead for a while yet, at least until accepted levels of security go beyond the AES equivalent of 384 bits, or perhaps until even better curve families are found.

6.11 Summary of contributions

This chapter presents an in depth analysis of eight attractive families that have not yet been examined in the context of defining implementation-friendly subfamilies. We have provided implementors with comprehensive lists of low Hamming

weight curves belonging to our favourite choices from each family, all of which offer highly efficient pairing instantiations. Our approach also facilitates implementors to easily define their own searches that target requirements they desire in any particular family.

Chapter 7

Hyperelliptic arithmetic via linear algebra

This chapter presents a novel method of divisor composition on arbitrary hyperelliptic curves. Whilst the algorithm we propose can immediately be used for pairing computation on a (pairing-friendly) hyperelliptic curve of any genus, its application currently has a greater impact outside the context of PBC. This is in large part due to the fact that ordinary pairing-friendly hyperelliptic curves of genus $g \geq 2$ and with ρ-value low enough to compete against their genus 1 counterparts have not yet been found. Thus, for this chapter only our discussion will assume a broader context than just that of PBC, but we reiterate that the application of our results to hyperelliptic pairings is inherent. Specifically, the functions that are needed in Miller's algorithm for hyperelliptic pairings come for free in our algorithm. For excellent surveys on the arena of hyperelliptic pairings, we refer to the works of Galbraith, Hess and Vercauteren [GHV07], and of Balakrishnan *et al.* [BBC$^+$09].

We derive an explicit method of computing the composition step in Cantor's algorithm for group operations on Jacobians of hyperelliptic curves. Our technique is inspired by the geometric description of the group law and applies to hyperelliptic curves of arbitrary genus. While Cantor's general composition involves arithmetic in the polynomial ring $\mathbb{F}_q[x]$, the algorithm we propose solves a linear system over the base field which can be written down directly from the Mumford coordinates of the group elements.

We apply this method to give more efficient formulas for group operations in both affine and projective coordinates for cryptographic systems based on Jacobians of genus 2 hyperelliptic curves in general form.

7.1 Motivation

In 1989, Koblitz [Kob89] generalised the idea of ECC by proposing Jacobians of hyperelliptic curves of arbitrary genus as a way to construct Abelian groups suitable for cryptography. Roughly speaking, hyperelliptic curves of genus g can achieve groups of the same size and security as elliptic curves, whilst being defined over finite fields with g times fewer bits[1]. At the same time however, increasing the genus of a hyperelliptic curve significantly increases the computational cost of performing a group operation in the corresponding Jacobian group. Thus, the question that remains of great interest to the public-key cryptography community is, under which circumstances elliptic curves are preferable, and vice versa. At the present time, elliptic curves carry on standing as the front-runner in most practical scenarios, but whilst both ECC and hyperelliptic curve cryptography (HECC) continue to enjoy a wide range of improvements, this question remains open in general. For a nice overview of the progress in this race and of the state-of-the-art in both cases, the reader is referred to the talks by Bernstein [Ber06], and by Lange [Lan06].

Cantor [Can87] was the first to give a concrete algorithm for performing computations in Jacobian groups of hyperelliptic curves over fields of odd characteristic. Shortly after, Koblitz [Kob89] modified this algorithm to apply to fields of any characteristic. Cantor's algorithm makes use of the polynomial representation of group elements proposed by Mumford [Mum84], and consists of two stages: (i) the *composition* stage, based on Gauss's classical composition of binary quadratic forms, which generally outputs an unreduced divisor, and (ii) the *reduction* stage, which transforms the unreduced divisor into the unique reduced divisor that is equivalent to the sum, whose existence is guaranteed by the Riemann-Roch theorem [Lan72]. Cantor's algorithm has since been substantially optimised in work initiated by Harley [Har], who was the first to obtain practical explicit formulas in genus 2, and extended by Lange [Lan01, Lan05], who, among

[1]The security argument becomes more complicated once venturing beyond genus 2, where the attacks by Gaudry [Gau00] and others [Die06, GTTD07, Smi09] overtake the Pollard Rho method [Pol78].

several others [MCT01, Tak02, MDM+02, SMCT02], generalised and significantly improved Harley's original approach. Essentially, all of these improvements involve unrolling the polynomial arithmetic implied by Cantor's algorithm into operations in the underlying field, and finding specialized shortcuts dedicated to each of the separate cases of input (see [Lan02a, §4]).

In this work we propose an explicit alternative to unrolling Cantor's polynomial arithmetic in the composition phase. Our method is inspired by considering the geometric description of the group law and applies to hyperelliptic curves of any genus. The equivalence of the geometric group law and Cantor's algorithm was proved by Lauter [Lau03] in the case of genus 2, but since then there have been almost no reported improvements in explicit formulas that benefit from this depiction, the notable exception being the work of Leitenberger [Lei05], who used Gröbner basis reduction to show that in the addition of two distinct divisors on the Jacobian of a genus 2 curve, one can obtain explicit formulas to compute the required geometric function directly from the Mumford coordinates without (unrolling) polynomial arithmetic. Leitenberger's idea of obtaining the necessary geometric functions in a simple and elementary way is central to the theme of our work in this chapter, although we note that the affine addition formulas that result from our description (which do not rely on any Gröbner basis reduction) are significantly faster than the direct translation of those given in [Lei05].

We use the geometric description of the group law to prove that the interpolating functions for the composition step can be found by writing down a linear system in the ground field to be solved in terms of the Mumford coordinates of the divisors. Therefore, the composition algorithm for arbitrary genera proposed in this work is immediately explicit in terms of arithmetic in $\mathbb{F}_q$, in contrast to Cantor's composition which operates in the polynomial ring $\mathbb{F}_q[x]$, the optimisation of which calls for ad-hoc attention in each genus to unravel the $\mathbb{F}_q[x]$ operations into explicit formulas in $\mathbb{F}_q$.

To illustrate the value of our approach, we show that, for group operations on Jacobians of general genus 2 curves over large prime fields, the (affine and projective) formulas that result from this description are more efficient than their predecessors. Also, when applying this approach back to the case of genus 1, we are able to recover several of the tricks previously explored for merging simultaneous group operations to optimise elliptic curve computations.

The rest of this chapter is organised as follows. We briefly touch on some

more related work below, before moving to Section 7.2 where we give a short background on hyperelliptic curves and the Mumford representation of Jacobian elements; this ties back to Section 2.2 where we defined the divisor class group. Section 7.3 discusses the geometry of Jacobian arithmetic on hyperelliptic curves, and shows that we can use simple linear algebra to compute the required geometric functions from the Mumford coordinates. Section 7.4 is dedicated to illustrating how this technique results in fast explicit formulas in genus 2, whilst Section 7.5 generalises the algorithm for all $g \geq 2$. As we hope this work will influence further progress in higher genus arithmetic, in Section 7.6 we highlight some further implications of adopting this geometrically inspired approach. Magma scripts that verify our proposed algorithms and formulas can be found in the full version of our original work[2].

More related work. There are several high-level papers (e.g. [HI94, Hes02]) which discuss general methods for computing in Jacobians of arbitrary algebraic curves. In addition, there has also been work which specifically addresses arithmetic on non-hyperelliptic Jacobians from a geometric perspective (e.g. [FOR08, FO04]).

Khuri-Makdisi treated divisor composition on arbitrary algebraic curves with linear algebra techniques in [KM04] and [KM07]. In contrast to Khuri-Makdisi's deep and more general approach, our work specifically aims to present an explicit algorithm in an implementation-ready format that is specific to hyperelliptic curves, much like his joint work with Abu Salem which applied his earlier techniques to present explicit formulas for arithmetic on $C_{3,4}$ curves [AKM06]. Some other authors have also applied techniques from the realm of linear algebra to Jacobian operations: two notable examples being the work of Guyot *et al.* [GKP04] and Avanzi *et al.* [ATW08] who both used matrix methods to compute the resultant of two polynomials in the composition stage.

Since we have focused on general hyperelliptic curves, our comparison in genus 2 does not include the record-holding work by Gaudry [Gau07], which exploits the Kummer surface associated with curves of a special form to achieve the current outright fastest genus 2 arithmetic for those curve models. Gaudry and Harley's second exposition [GH00] further describes the results in [Har]. Finally, we do not draw comparisons with any work on real models of hyperelliptic curves, which

[2] See `http://eprint.iacr.org/2011/306`

usually result in slightly slower formulas than imaginary hyperelliptic curves, but we note that both Galbraith *et al.* [GHM08] and Erickson *et al.* [EJSS10] achieve very competitive formulas for group law computations on real models of genus 2 hyperelliptic curves.

7.2 The Mumford representation

We revisit some of the discussion from Section 2.2 before defining the Mumford representation of points in the Jacobian. For a more in depth discussion, we refer to [ACD+05, §4] and [Gal12, §11]. Recall from Equation (2.13) that over the field K, we use C_g to denote the general ("imaginary quadratic") hyperelliptic curve of genus g given by

$$\begin{gathered} C_g : y^2 + h(x)y = f(x), \\ h(x), f(x) \in K[x], \quad \deg(f) = 2g+1, \quad \deg(h) \leq g, \quad f \text{ monic}, \end{gathered} \tag{7.1}$$

with the added stipulation that no point $(x, y) \in \overline{K}$ simultaneously sends both partial derivatives $2y + h(x)$ and $f'(x) - h'(x)y$ to zero [ACD+05, §14.1]. As long as $\mathrm{char}(K) \neq 2g+1$, we can isomorphically transform C_g into $\hat{C}_g$, given as $\hat{C}_g : y^2 + h(x)y = x^{2g+1} + \hat{f}_{2g-1}x^{2g-1} + ... + \hat{f}_1 x + \hat{f}_0$, so that the coefficient of x^{2g} is zero [ACD+05, §14.13]. In the case of odd characteristic fields, it is standard to also annihilate the presence of $h(x)$ completely under a suitable transformation, in order to obtain a simpler model (we will make use of this in §7.4). We abuse notation and use C_g from hereon to refer to the simplified version of the curve equation in each context. Although the proofs in §7.3 apply to any K, it better places the intention of the discussion to henceforth regard K as a finite field $\mathbb{F}_q$.

We work in the Jacobian group $\mathrm{Jac}(C_g)$ of C_g, where the elements are equivalence classes of degree zero divisors on C_g – i.e. the reader can identify the Jacobian with the Picard group $\mathrm{Div}^0_K(C_g)$ from Section 2.2 (but see [GHV07, §2.1] for the precise definition). It follows from the Riemann-Roch Theorem (see Section 2.2.2) that for hyperelliptic curves, each class D has a unique *reduced* representative of the form

$$\rho(D) = (P_1) + (P_2) + ... + (P_r) - r(P_\infty),$$

such that $r \leq g$, $P_i \neq -P_j$ for $i \neq j$, no P_i satisfying $P_i = -P_i$ appears more

than once, and with P_∞ being the point at infinity on C_g. We drop the ρ from hereon and, unless stated otherwise, assume divisor equations involve reduced divisors. When referring to the non-trivial elements in the reduced divisor D, we mean all $P \in \mathrm{supp}(D)$ where $P \neq P_\infty$, i.e. the elements corresponding to the effective part of D. For each of the r non-trivial elements appearing in D, write $P_i = (x_i, y_i)$. Mumford proposed a convenient way to represent such divisors as $D = (u(x), v(x))$, where $u(x)$ is a monic polynomial with $\deg(u(x)) \leq g$ satisfying $u(x_i) = 0$, and $v(x)$ (which is not monic in general) with $\deg(v(x)) < \deg(u(x))$ is such that $v(x_i) = y_i$, for $1 \leq i \leq r$. In this way we have a one-to-one correspondence between reduced divisors and their so-called *Mumford representation* [Mum84]. In this chapter we return to the notation $\oplus$ (resp. $\ominus$) to distinguish group additions (resp. subtractions) between Jacobian elements from "additions" in formal divisor sums - this is because we now assume that more than one point is contained in the effective part of D. We use $\bar{D}$ to denote the divisor obtained by taking the hyperelliptic involution of each of the non-trivial elements in the support of D.

When developing formulas for implementing genus g arithmetic, we are largely concerned with the frequent case that arises where both (not necessarily distinct) reduced divisors $D = (u(x), v(x))$ and $D' = (u'(x), v'(x))$ in the sum $D \oplus D'$ are such that $\deg(u(x)) = \deg(u'(x)) = g$. This means that $D = E - g(P_\infty)$ and $D' = E' - g(P_\infty)$, with both E and E' being effective divisors of degree g; from hereon we interchangeably refer to such divisors D as *full degree* or *degree* g divisors, and we use $\hat{\mathrm{J}}\mathrm{ac}(C_g)$ to denote the set of all such divisor classes of full degree, where $\hat{\mathrm{J}}\mathrm{ac}(C_g) \subset \mathrm{Jac}(C_g)$. In Section 7.5.2 we discuss how to handle the special case when a divisor of degree less than g is encountered.

7.3 Computations in the Mumford function field

The purpose of this section is to show how to compute group law operations in Jacobians by applying linear algebra to the Mumford coordinates of divisors. The geometric description of the group law is an important ingredient in the proof of the proposed linear algebra approach (particularly in the proof of Proposition 7.5), so we start by reviewing the geometry underlying arithmetic on Jacobians of hyperelliptic curves.

Since the Jacobian of a hyperelliptic curve is the group of degree zero divisors

modulo principal divisors, the group operation is formal addition modulo the equivalence relation. Thus two divisors D and D' can be added by finding a function whose divisor contains the support of both D and D', and then the sum is equivalent to the negative of the complement of that support. Such a function $\ell(x)$ can be obtained by interpolating the points in the support of the two divisors. The complement of the support of D and D' in the support of $\mathrm{div}(\ell)$ consists of the other points of intersection of ℓ with the curve. In general those individual points may not be defined over the ground field for the curve. We are thus led to work with Mumford coordinates for divisors on hyperelliptic curves, since the polynomials in Mumford coordinates are defined over the base field and allow us to avoid extracting individual roots and working with points defined over extension fields.

For example, consider adding two full degree genus 3 divisors $D, D' \in \hat{\mathrm{J}}\mathrm{ac}(C_3)$, with respective supports $\mathrm{supp}(D) = \{P_1, P_2, P_3\} \cup \{P_\infty\}$ and $\mathrm{supp}(D') = \{P'_1, P'_2, P'_3\} \cup \{P_\infty\}$, as in Figure 7.1 and Figure 7.2.

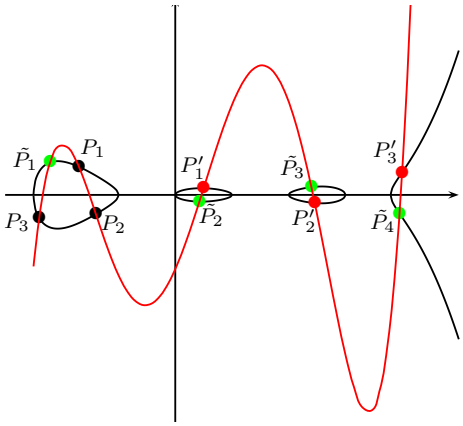


Figure 7.1: The *composition* stage of a general addition on the Jacobian of a genus 3 curve C_3 over the reals $\mathbb{R}$: the 6 points in the combined supports of D and D' are interpolated by a quintic polynomial which intersects C in 4 more places to form the unreduced divisor $\tilde{D} = \tilde{P}_1 + \tilde{P}_2 + \tilde{P}_3 + \tilde{P}_4$.

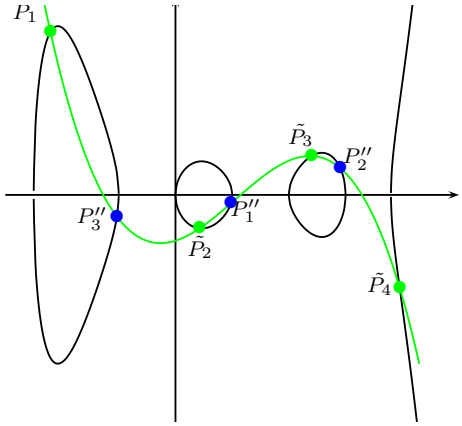


Figure 7.2: The *reduction* stage: a (vertically) magnified view of the cubic function which interpolates the points in the support of $\tilde{D}$ and intersects C_3 in three more places to form $\bar{D}'' = (P''_1 + P''_2 + P''_3) \sim \tilde{D}$, the reduced equivalent of $\tilde{D}$.

After computing the quintic function $\ell(x, y) = \sum_{i=0}^{5} \ell_i x^i$ that interpolates

the six non-trivial points in the composition phase, computing the x-coordinates of the remaining (four) points of intersection explicitly would require solving

$$\ell_5^2 \cdot \prod_{i=1}^{3}(x - x_i) \cdot \prod_{i=1}^{3}(x - x_i') \prod_{i=1}^{4}(x - \bar{x}_i) = \big(\sum_{i=0}^{5} \ell_i x^i\big)^2 - f(x)$$

for $\bar{x}_1$,$\bar{x}_2$,$\bar{x}_3$ and $\bar{x}_4$, which would necessitate multiple root extractions. On the other hand, the exact division $\prod_{i=1}^{4}(x - \bar{x}_i) = \Big(\big(\sum_{i=0}^{5} \ell_i x^i\big)^2 - f(x)\Big) / \Big(\ell_5^2 \cdot \prod_{i=1}^{3}(x - x_i) \cdot \prod_{i=1}^{3}(x - x_i')\Big)$ can be computed very efficiently (and entirely over $\mathbb{F}_q$) by equating coefficients of x.

Whilst the Mumford representation is absolutely necessary for efficient reduction, the price we seemingly pay in deriving formulas from the simple geometric description lies in the composition phase. In any case, finding the interpolating function $y = \ell(x)$ would be conceptually trivial if we knew the (x, y) coordinates of the points involved, but computing the function directly from the Mumford coordinates appears to be more difficult. In what follows we detail how this can be achieved in general, using only linear algebra over the base field. The meanings of the three propositions in this section are perhaps best illustrated through the examples that follow each of them.

Proposition 7.1. *On the Jacobian of a genus g hyperelliptic curve, the dense set $\hat{\mathrm{J}}\mathrm{ac}(C_g)$ of divisor classes with reduced representatives of full degree g can be described exactly as the intersection of g hypersurfaces of dimension (at most) $2g$.*

Proof. Let $D = \big(u(x), v(x)\big) = \big(x^g + \sum_{i=0}^{g-1} u_i x^i \,,\, \sum_{i=0}^{g-1} v_i x^i\big) \in \hat{\mathrm{J}}\mathrm{ac}(C_g(K))$ be an arbitrary degree g divisor class representative with $\mathrm{supp}(D) = \{(x_1, y_1), ..., (x_g, y_g)\} \cup \{P_\infty\}$, so that $u(x_i) = 0$ and $v(x_i) = y_i$ for $1 \leq i \leq g$. Let $\Psi(x) = \sum_{i=0}^{g-1} \Psi_i x^i$ be the polynomial obtained by substituting $y = v(x)$ into the equation for C_g and reducing modulo the ideal generated by $u(x)$. Clearly, $\Psi(x_i) \equiv 0 \bmod \langle u(x) \rangle$ for each of the g non-trivial elements in $\mathrm{supp}(D)$, but since $\deg(\Psi(x)) \leq g-1$, it follows that each of its g coefficients Ψ_i must be identically zero, implying that every element $D \in \hat{\mathrm{J}}\mathrm{ac}(C_g)$ of full degree g lies in the intersection of the g hypersurfaces $\Psi_i = \Psi_i(u_0, ..., u_{g-1}, v_0, ..., v_{g-1}) = 0$. On the other hand, each unique $2g$-tuple in K which satisfies $\Psi_i = 0$ for $1 \leq i \leq g$ defines a unique full degree representative $D \in \hat{\mathrm{J}}\mathrm{ac}(C_g(K))$ (cf. [Gal12, ex 11.3.7]). □

Definition 7.2 (Mumford ideals)**.** *We call the g ideals* $\langle \Psi_i \rangle$ *arising from the g hypersurfaces* $\Psi_i = 0$ *in Proposition 7.1 the Mumford ideals.*

Definition 7.3 (Mumford function fields)**.** *The function fields of* $\hat{\mathrm{J}}\mathrm{ac}(C_g)$ *and* $\hat{\mathrm{J}}\mathrm{ac}(C_g) \times \hat{\mathrm{J}}\mathrm{ac}(C_g)$ *are respectively identified with the quotient fields of*

$$\frac{K[u_0, ..., u_{g-1}, v_0, ..., v_{g-1}]}{\langle \Psi_0, ..., \Psi_{g-1} \rangle} \text{ and } \frac{K[u_0, ..., u_{g-1}, v_0, ..., v_{g-1}, u'_0, ..., u'_{g-1}, v'_0, ..., v'_{g-1}]}{\langle \Psi_0, ..., \Psi_{g-1}, \Psi'_0, ..., \Psi'_{g-1} \rangle},$$

which we call the Mumford function fields and denote by $K^{\mathrm{Mum}}_{\mathrm{DBL}} = K(\hat{\mathrm{J}}\mathrm{ac}(C_g))$ *and* $K^{\mathrm{Mum}}_{\mathrm{ADD}} = K(\hat{\mathrm{J}}\mathrm{ac}(C_g) \times \hat{\mathrm{J}}\mathrm{ac}(C_g))$ *respectively. We abbreviate and use* Ψ_i, Ψ'_i *to differentiate between* $\Psi_i = \Psi_i(u_0, ..., u_{g-1}, v_0, ..., v_{g-1})$ *and* $\Psi'_i = \Psi_i(u'_0, ..., u'_{g-1}, v'_0, ..., v'_{g-1})$ *when working in* $K^{\mathrm{Mum}}_{\mathrm{ADD}}$*.*

Example 7.3.1. Consider the genus 2 hyperelliptic curve defined by $C : y^2 = (x^5 + 2x^3 - 7x^2 + 5x + 1)$ over $\mathbb{F}_{37}$. A general degree two divisor $D \in \hat{\mathrm{J}}\mathrm{ac}(C)$ takes the form $D = (x^2 + u_1 x + u_0, v_1 x + v_0)$. Substituting $y = v_1 x + v_0$ into C and reducing modulo $\langle x^2 + u_1 x + u_0 \rangle$ gives

$$(v_1 x + v_0)^2 - (x^5 + 2x^3 - 7x^2 + 5x + 1) \equiv \Psi_1 x + \Psi_0 \equiv 0 \mod \langle x^2 + u_1 x + u_0 \rangle$$

where

$$\Psi_1(u_1, u_0, v_1, v_0) = 3\, u_0 {u_1}^2 - {u_1}^4 - {u_0}^2 + 2\, v_0 v_1 - {v_1}^2 u_1 + 2\,(u_0 - {u_1}^2) - 7u_1 - 5,$$
$$\Psi_0(u_1, u_0, v_1, v_0) = {v_0}^2 - {v_1}^2 u_0 + 2\, {u_0}^2 u_1 - {u_1}^3 u_0 - 2u_1 u_0 - 7u_0 - 1.$$

The number of tuples $(u_0, u_1, v_0, v_1) \in \mathbb{F}_{37}$ lying in the intersection of $\Psi_0 = \Psi_1 = 0$ is 1373, which is the number of degree 2 divisors on $\mathrm{Jac}(C)$, i.e. $\#\hat{\mathrm{J}}\mathrm{ac}(C) = 1373$. There are 39 other divisors on $\mathrm{Jac}(C)$ with degrees less than 2, each of which is isomorphic to a point on the curve, so that $\#\mathrm{Jac}(C) = \#\hat{\mathrm{J}}\mathrm{ac}(C) + \#C = 1373 + 39 = 1412$. Formulas for performing full degree divisor additions are derived inside the Mumford function field $K^{\mathrm{Mum}}_{\mathrm{ADD}} = \mathrm{Quot}(K[u_0, u_1, v_0, v_1, u'_0, u'_1, v'_0, v'_1] / \langle \Psi_0, \Psi_1, \Psi'_0, \Psi'_1 \rangle)$, whilst formulas for full degree divisor doublings are derived inside the Mumford function field $K^{\mathrm{Mum}}_{\mathrm{DBL}} = \mathrm{Quot}(K[u_0, u_1, v_0, v_1] / \langle \Psi_0, \Psi_1 \rangle)$.

Performing the efficient composition of two divisors amounts to finding the least degree polynomial function that interpolates the union of their (assumed disjoint) non-trivial supports. The following two propositions show that in the

general addition and doubling of divisors, finding the interpolating functions in the Mumford function fields can be accomplished by solving linear systems.

Proposition 7.4 (General divisor addition)**.** *Let D and D' be reduced divisors of degree g on* $\mathrm{Jac}(C_g)$ *such that* $\mathrm{supp}(D) = \{(x_1, y_1), ..., (x_g, y_g)\} \cup \{P_\infty\}$, $\mathrm{supp}(D') = \{(x'_1, y'_1), ..., (x'_g, y'_g)\} \cup \{P_\infty\}$ *and $x_i \neq x'_j$ for all $1 \leq i, j \leq g$. A function ℓ on C_g that interpolates the $2g$ non-trivial elements in* $\mathrm{supp}(D) \cup \mathrm{supp}(D')$ *can be determined by solving a linear system of dimension $2g$ inside the Mumford function field $K_{\mathrm{ADD}}^{\mathrm{Mum}}$.*

Proof. Let $D = \big(u(x), v(x)\big) = \big(x^g + \sum_{i=0}^{g-1} u_i x^i \,,\, \sum_{i=0}^{g-1} v_i x^i\big)$ and $D' = \big(u'(x), v'(x)\big) = \big(x^g + \sum_{i=0}^{g-1} u'_i x^i \,,\, \sum_{i=0}^{g-1} v'_i x^i\big)$. Let the polynomial $y = \ell(x) = \sum_{i=0}^{2g-1} \ell_i x^i$ be the desired function that interpolates the $2g$ non-trivial elements in $\mathrm{supp}(D) \cup \mathrm{supp}(D')$, i.e. $y_i = \ell(x_i)$ and $y'_i = \ell(x'_i)$ for $1 \leq i \leq g$. Focussing firstly on D, it follows that $v(x) - \ell(x) = 0$ for $x \in \{x_i\}_{1 \leq i \leq g}$. As in the proof of Proposition 7.1, we reduce modulo the ideal generated by $u(x)$ giving $\Omega(x) = v(x) - \ell(x) \equiv \sum_{i=0}^{g-1} \Omega_i x^i \equiv 0 \mod \langle x^g + \sum_{i=0}^{g-1} u_i x^i \rangle$. Since $\deg(\Omega(x)) \leq g - 1$ and $\Omega(x_i) = 0$ for $1 \leq i \leq g$, it follows that the g coefficients $\Omega_i = \Omega_i(u_0, ..., u_{g-1}, v_0, ..., v_{g-1}, \ell_0, ..., \ell_{2g-1})$ must be all identically zero. Each gives rise to an equation that relates the $2g$ coefficients of $\ell(x)$ linearly inside $K_{\mathrm{ADD}}^{\mathrm{Mum}}$. Defining $\Omega'(x)$ from D' identically and reducing modulo $u'(x)$ gives another g linear equations in the $2g$ coefficients of $\ell(x)$. □

Example 7.3.2. Consider the genus 3 hyperelliptic curve defined by $C : y^2 = x^7 + 1$ over $\mathbb{F}_{71}$, and take $D = \big(u(x), v(x)\big), D' = \big(u'(x), v'(x)\big) \in \hat{\mathrm{J}}\mathrm{ac}(C)$ as

$$D = \big(x^3 + 6x^2 + 41x + 33, 29x^2 + 22x + 47\big),$$
$$D' = \big(x^3 + 18x^2 + 15x + 37, 49x^2 + 46x + 59\big).$$

We compute the polynomial $\ell(x) = \sum_{i=0}^{5} \ell_i x^i$ that interpolates the six non-trivial elements in $\mathrm{supp}(D) \cup \mathrm{supp}(D')$ using $\ell(x) - v(x) \equiv 0 \bmod \langle u(x) \rangle$ and $\ell(x) - v'(x) \equiv 0 \bmod \langle u'(x) \rangle$, to obtain Ω_i and Ω'_i for $0 \leq i \leq 2$. For D and D', we respectively have that

$$0 \equiv \ell(x) - (29x^2 + 22x + 47) \equiv \Omega_2 x^2 + \Omega_1 x + \Omega_0 \bmod \langle x^3 + 6x^2 + 41x + 33 \rangle,$$
$$0 \equiv \ell(x) - (49x^2 + 46x + 59) \equiv \Omega'_2 x^2 + \Omega'_1 x + \Omega'_0 \bmod \langle x^3 + 18x^2 + 15x + 37 \rangle,$$

with

$$\Omega_2 = \ell_2 + 65\ell_3 + 66\ell_4 + 30\ell_5 - 29; \qquad \Omega_2' = \ell_2 + 53\ell_3 + 25\ell_4 + 67\ell_5 - 49;$$
$$\Omega_1 = \ell_1 + 30\ell_3 + 48\ell_5 - 22; \qquad \Omega_1' = \ell_1 + 56\ell_3 + 20\ell_4 + 7\ell_5 - 46;$$
$$\Omega_0 = \ell_0 + 38\ell_3 + 56\ell_4 + 23\ell_5 - 47; \qquad \Omega_0' = \ell_0 + 34\ell_3 + 27\ell_4 + 69\ell_5 - 59.$$

Solving $\Omega_{0\leq i\leq 2}, \Omega'_{0\leq i\leq 2} = 0$ simultaneously for $\ell_0, ..., \ell_5$ gives $\ell(x) = 21x^5 + x^4 + 36x^3 + 46x^2 + 64x + 57$.

Proposition 7.5 (General divisor doubling)**.** *Let D be a divisor of degree g representing a class on* $\mathrm{Jac}(C_g)$ *with* $\mathrm{supp}(D) = \{P_1, ..., P_g\} \cup \{P_\infty\}$*. A function ℓ on C_g such that each non-trivial element in* $\mathrm{supp}(D)$ *occurs with multiplicity two in* $\mathrm{div}(\ell)$ *can be determined by a linear system of dimension $2g$ inside the Mumford function field $K_{\mathrm{DBL}}^{\mathrm{Mum}}$.*

Proof. Let $D = \big(u(x), v(x)\big) = \big(\, x^g + \sum_{i=0}^{g-1} u_i x^i \,,\, \sum_{i=0}^{g-1} v_i x^i \,\big)$ and write $P_i = (x_i, y_i)$ for $1 \leq i \leq g$. Let the polynomial $y = \ell(x) = \sum_{i=0}^{2g-1} \ell_i x^i$ be the desired function that interpolates the g non-trivial elements of $\mathrm{supp}(D)$, and also whose derivative $\ell'(x)$ is equal to dy/dx on $C_g(x, y)$ at each such element. Namely, $\ell(x) = \sum_{i=0}^{2g-1} \ell_i x^i$ is such that $\ell(x_i) = y_i$ and $\frac{d\ell}{dx}(x_i) = \frac{dy}{dx}(x_i)$ on C for $1 \leq i \leq g$. This time the first g equations come from the direct interpolation as before, whilst the second g equations come from the general expression for the equated derivates, taking $\frac{d\ell}{dx}(x_i) = \frac{dy}{dx}(x_i)$ on C_g as

$$\sum_{i=1}^{g-1} i\ell_i x^{i-1} = \frac{(2g+1)x^{2g} + \sum_{i=1}^{2g-1} i f_i x^{i-1} + (\sum_{i=0}^{g} i h_i x^{i-1}) \cdot y}{2y + \sum_{i=0}^{g} h_i x^i}$$

for each x_i with $1 \leq i \leq g$. Again, it is easy to see that substituting $y = v(x)$ and reducing modulo the ideal generated by $u(x)$ will produce a polynomial $\Omega'(x)$ with degree less than or equal to $g - 1$. Since $\Omega'(x)$ has g roots, $\Omega'_i = 0$ for $0 \leq i \leq g - 1$, giving rise to the second g equations which importantly relate the coefficients of $\ell(x)$ linearly inside $K_{\mathrm{DBL}}^{\mathrm{Mum}}$. □

Example 7.3.3. Consider the genus 3 hyperelliptic curve defined by $C : y^2 = x^7 + 5x + 1$ over $\mathbb{F}_{257}$, and take $D \in \hat{\mathrm{J}}\mathrm{ac}(C)$ as $D = (u(x), v(x)) = (x^3 + 57x^2 + 26x + 80, 176x^2 + 162x + 202)$. We compute the polynomial $\ell(x) = \sum_{i=0}^{5} \ell_i x^i$ that interpolates the three non-trivial points in $\mathrm{supp}(D)$, and also has the same

derivative as C at these points. For the interpolation only, we obtain $\Omega_0, \Omega_1, \Omega_2$ (collected below) identically as in Example 7.3.2.

For $\Omega_0', \Omega_1', \Omega_2'$, equating dy/dx on C with $\ell'(x)$ gives

$$\frac{7x^6+5}{2y} \equiv 5\ell_5x^4 + 4\ell_4x^3 + 3\ell_3x^2 + 2\ell_2x + \ell_1 \mod \langle x^3 + 57x^2 + 26x + 80\rangle,$$

which, after substituting $y = 176x^2 + 162x + 202$, rearranges to give $0 \equiv \Omega_2'x^2 + \Omega_1'x + \Omega_0'$, where

$$\begin{aligned}
\Omega_2 &= 118\ell_4 + 256\ell_2 + 57\ell_3 + 96\ell_5; & \Omega_2' &= 76\ell_5 + 2541\ell_4 + 254\ell_3 + 166;\\
\Omega_1 &= 140\ell_4 + 256\ell_1 + 26\ell_3 + 82\ell_5; & \Omega_1' &= 209 + 255\ell_2 + 104\ell_4 + 186\ell_5;\\
\Omega_0 &= 256\ell_0 + 80\ell_3 + 69\ell_5 + 66\ell_4; & \Omega_0' &= 73\ell_5 + 63\ell_4 + 256\ell_1 + 31.
\end{aligned}$$

Solving $\Omega_{0\le i\le 2}, \Omega'_{0\le i\le 2} = 0$ simultaneously for $\ell_0, ..., \ell_5$ gives $\ell(x) = 84x^5+213x^3+78x^2+252x+165$.

This section showed that divisor composition on hyperelliptic curves can be achieved via linear operations in the Mumford function fields.

7.4 Generating explicit formulas in genus 2

This section applies the results of the previous section to develop explicit formulas for group law computations involving full degree divisors on Jacobians of genus 2 hyperelliptic curves. Assuming an underlying field of large prime characteristic, such genus 2 hyperelliptic curves $C'/\mathbb{F}_q$ can always be isomorphically transformed into $C_2/\mathbb{F}_q$ given by $C_2 : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0$, where $C_2 \cong C'$ (see §7.2). The Mumford representation of a general degree two divisor $D \in \hat{\mathrm{J}}\mathrm{ac}(C_2) \subset \mathrm{Jac}(C_2)$ is given as $D = (x^2 + u_1x + u_0, v_1x + v_0)$. From Proposition 7.1, we compute the $g = 2$ hypersurfaces whose intersection is the set of all such divisors $\hat{\mathrm{J}}\mathrm{ac}(C_2)$ as follows. Substituting $y = v_1x + v_0$ into the equation for C_2 and reducing modulo the ideal $\langle x^2 + u_1x + u_0\rangle$ gives the polynomial $\Psi(x)$ as

$$\Psi(x) \equiv \Psi_1x + \Psi_0 \equiv (v_1x + v_0)^2 - (x^5 + f_3x^3 + f_2x^2 + f_1x + f_0) \mod \langle x^2 + u_1x + u_0\rangle,$$

where

$$\Psi_0 = {v_0}^2 - f_0 + f_2 u_0 - {v_1}^2 u_0 + 2\,{u_0}^2 u_1 - u_1 f_3 u_0 - {u_1}^3 u_0,$$
$$\Psi_1 = 2\,v_0 v_1 - f_1 - {v_1}^2 u_1 + f_2 u_1 - f_3({u_1}^2 - u_0) + 3\,u_0 {u_1}^2 - {u_1}^4 - {u_0}^2. \quad (7.2)$$

We will derive doubling formulas inside $K_{\mathrm{ADD}}^{\mathrm{Mum}} = \mathrm{Quot}(K[u_0, u_1, v_0, v_1]/\langle \Psi_0, \Psi_1 \rangle)$ and addition formulas inside $K_{\mathrm{ADD}}^{\mathrm{Mum}} = \mathrm{Quot}(K[u_0, u_1, v_0, v_1, u_0', u_1', v_0', v_1']$ $/\langle \Psi_0, \Psi_1, \Psi_0', \Psi_1' \rangle)$. In §7.4.2 particularly, we will see how the ideal $\langle \Psi_0, \Psi_1 \rangle$ is useful in simplifying the formulas that arise.

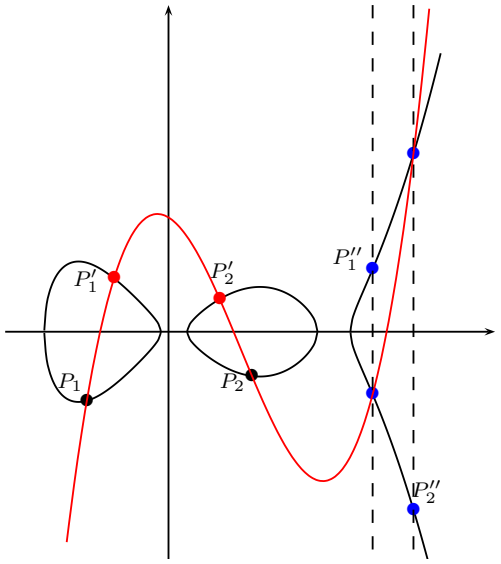


Figure 7.3: The group law (general addition) on the Jacobian of the genus 2 curve C_2 over the reals $\mathbb{R}$, for $(P_1 + P_2) \oplus (P_1' + P_2') = P_1'' + P_2''$.

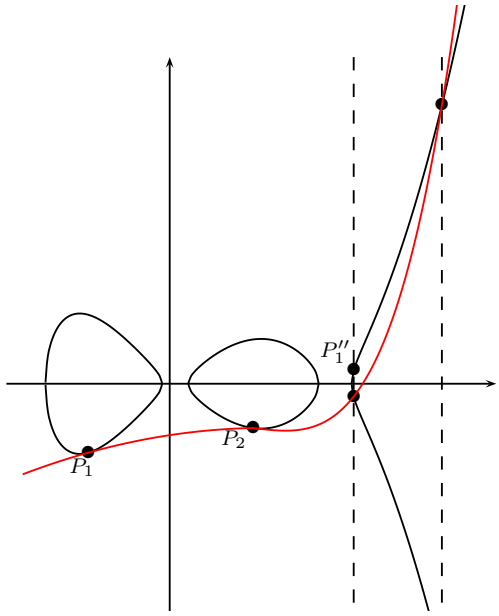


Figure 7.4: A general point doubling on the Jacobian of a genus 2 curve C_2 over the reals $\mathbb{R}$, for $[2](P_1+P_2) = P_1'' + P_2''$.

7.4.1 General divisor addition in genus 2

Let $D = (x^2 + u_1 x + u_0, v_1 x + v_0), D' = (x^2 + u_1' x + u_0', v_1' x + v_0') \in \hat{\mathrm{J}}\mathrm{ac}(C_2)$ be two divisors with $\mathrm{supp}(D) = \{P_1, P_2\} \cup \{P_\infty\}$ and $\mathrm{supp}(D') = \{P_1', P_2'\} \cup \{P_\infty\}$, such that no P_i has the same x coordinate as P_j' for $1 \leq i, j \leq 2$. Let $D'' = (x^2 + u_1'' x + u_0'', v_1'' x + v_0'') = D \oplus D'$. The composition step in the addition of D and D' involves building the linear system inside $K_{\mathrm{ADD}}^{\mathrm{Mum}}$ that solves to give the

coefficients ℓ_i of the cubic polynomial $y = \ell(x) = \sum_{i=0}^{3} \ell_i x^i$ which interpolates P_1, P_2, P_1', P_2'. Following Proposition 7.4, we have

$$\begin{aligned} 0 &\equiv \Omega_1 x + \Omega_0 \equiv \ell_3 x^3 + \ell_2 x^2 + \ell_1 x + \ell_0 - (v_1 x + v_0) \\ &\equiv (\ell_3({u_1}^2 - u_0) - \ell_2 u_1 + \ell_1 - v_1)x + (\ell_3 u_1 u_0 - \ell_2 u_0 + \ell_0 - v_0) \\ &\qquad \bmod \langle x^2 + u_1 x + u_0 \rangle \quad , \end{aligned} \tag{7.3}$$

which provides two equations ($\Omega_1 = 0$ and $\Omega_0 = 0$) relating the four coefficients of the interpolating polynomial linearly inside $K_{\mathrm{ADD}}^{\mathrm{Mum}}$. Identically, interpolating the support of D' produces two more linear equations which allow us to solve for the four ℓ_i as

$$\begin{pmatrix} 1 & 0 & -u_0 & u_1 u_0 \\ 0 & 1 & -u_1 & u_1^2 - u_0 \\ 1 & 0 & -u_0' & u_1' u_0' \\ 0 & 1 & -u_1' & {u_1'}^2 - u_0' \end{pmatrix} \cdot \begin{pmatrix} \ell_0 \\ \ell_1 \\ \ell_2 \\ \ell_3 \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_0' \\ v_1' \end{pmatrix}.$$

Observe that the respective subtraction of rows 1 and 2 from rows 3 and 4 gives rise to a smaller system that can be solved for ℓ_2 and ℓ_3, as

$$\begin{pmatrix} u_0 - u_0' & u_1' u_0' - u_1 u_0 \\ u_1 - u_1' & ({u_1'}^2 - u_0') - (u_1^2 - u_0) \end{pmatrix} \cdot \begin{pmatrix} \ell_2 \\ \ell_3 \end{pmatrix} = \begin{pmatrix} v_0' - v_0 \\ v_1' - v_1 \end{pmatrix}. \tag{7.4}$$

Remark 7.4.1. We will see in Section 7.5.1 that for all $g \geq 2$, the linear system that arises in the computation of $\ell(x)$ can always be trivially reduced to be of dimension g, but for now it is useful to observe that once we solve the dimension $g = 2$ matrix system for ℓ_i with $i \geq g$, calculating the remaining ℓ_i where $i < g$ is computationally straightforward.

The next step is to determine the remaining intersection points of $y = \ell(x)$ on C_2. Since $y = \ell(x)$ is cubic, its substitution into C_2 will give a degree six equation in x. Four of the roots will correspond to the four non-trivial points in $\mathrm{supp}(D) \cup \mathrm{supp}(D')$, whilst the remaining two will correspond to the two x coordinates of the non-trivial elements in $\mathrm{supp}(\bar{D}'')$, which are the same as the x coordinates in $\mathrm{supp}(D'')$ (see the intersection points in Figure 3). Let the Mumford representation of $\bar{D}''$ be $\bar{D}'' = (x^2 + {u_1}''x + {u_0}'', -v_1'' x - v_0'')$; we then

have

$$(x^2 + u_1 x + u_0) \cdot (x^2 + u_1' x + u_0') \cdot (x^2 + u_1'' x + u_0'') = \frac{(\sum_{i=0}^{3} \ell_i x^i)^2 - f(x)}{\ell_3^2}.$$

Equating coefficients is an efficient way to compute the exact division required above to solve for $u''(x)$. For example, equating coefficients of x^5 and x^4 above respectively gives

$$\begin{aligned} u_1'' &= -u_1 - u_1' - \frac{1 - 2\ell_2\ell_3}{\ell_3^2}; \\ u_0'' &= -(u_0 + u_0' + u_1 u_1' + (u_1 + u_1')u_1'') + \frac{2\ell_1\ell_3 + \ell_2^2}{\ell_3^2}. \end{aligned} \tag{7.5}$$

It remains to compute v_1'' and v_0''. Namely, we wish to compute the linear function that interpolates the points in $\mathrm{supp}(D'')$. Observe that reducing $\ell(x)$ modulo $\langle x^2 + u_1'' x + u_0'' \rangle$ gives the linear polynomial $-v_1'' x + -v_0''$ which interpolates the points in $\mathrm{supp}(\bar{D}'')$, i.e. those points which are the involutions of the points in $\mathrm{supp}(D'')$. Thus, the computation of v_1'' and v_0'' amounts to negating the result of $\ell(x)$ mod $\langle x^2 + u_1'' x + u_0'' \rangle$. From equation (7.3) then, it follows that

$$v_1'' = -(\ell_3(u_1''^2 - u_0'') - \ell_2 u_1'' + \ell_1), \qquad v_0'' = -(\ell_3 u_1'' u_0'' - \ell_2 u_0'' + \ell_0). \tag{7.6}$$

We summarise the process of computing a general addition $D'' = D \oplus D'$ on $\hat{\mathrm{J}}\mathrm{ac}(C_2)$, as follows. *Composition* involves constructing and solving the linear system in (7.4) for ℓ_2 and ℓ_3 before computing ℓ_0 and ℓ_1 via (7.3), whilst *reduction* involves computing u_1'' and u_0'' from (7.5) before computing v_1'' and v_0'' via (7.6). The explicit formulas for these computations are in Table 7.1, where **I**, **M** and **S** represent the costs of an $\mathbb{F}_q$ inversion, multiplication and squaring respectively. We postpone comparisons with other works until after the doubling discussion.

Remark 7.4.2. The formulas for computing v_0'' and v_1'' in (7.6) include operations involving $u_1''^2$ and $u_1'' u_0''$. Since those quantities are also needed in the first step of the addition formulas (see the first line of Table 7.1) for any subsequent additions involving the divisor D'', it makes sense to carry those quantities along as extra coordinates to exploit these overlapping computations. It turns out that an analogous overlap arises in geometric group operations for all $g \geq 2$, but for now we remark that both additions and doublings on genus 2 curves will benefit from extending the generic affine coordinate system to include two extra coordinates

AFFINE ADDITION			
Input:	$D = (u_1, u_0, v_1, v_0, U_1 = u_1^2, U_0 = u_1 u_0)$, $D' = (u_1', u_0', v_1', v_0', U_1' = u_1'^2, U_0' = u_1' u_0')$		Operations in $\mathbb{F}_q$
	$\sigma_1 \leftarrow u_1 + u_1', \quad \Delta_0 \leftarrow v_0 - v_0', \quad \Delta_1 \leftarrow v_1 - v_1', \quad M_1 \leftarrow U_1 - u_0 - U_1' + u_0', \quad M_2 \leftarrow U_0' - U_0,$		
	$M_3 \leftarrow u_1 - u_1', \quad M_4 \leftarrow u_0' - u_0, \quad t_1 \leftarrow (M_2 - \Delta_0) \cdot (\Delta_1 - M_1), \quad t_2 \leftarrow (-\Delta_0 - M_2) \cdot (\Delta_1 + M_1),$		2M
	$t_3 \leftarrow (-\Delta_0 + M_4) \cdot (\Delta_1 - M_3), \quad t_4 \leftarrow (-\Delta_0 - M_4) \cdot (\Delta_1 + M_3),$		2M
	$\ell_2 \leftarrow t_1 - t_2 \quad \ell_3 \leftarrow t_3 - t_4, \quad d \leftarrow t_3 + t_4 - t_1 - t_2 - 2(M_2 - M_4) \cdot (M_1 + M_3),$		1M
	$A \leftarrow 1/(d \cdot \ell_3), \quad B \leftarrow d \cdot A, \quad C \leftarrow d \cdot B, \quad D \leftarrow \ell_2 \cdot B, \quad E \leftarrow \ell_3^2 \cdot A, \quad CC \leftarrow C^2,$		$\mathbf{I} + 5\mathbf{M} + 2\mathbf{S}$
	$u_1'' \leftarrow 2D - CC - \sigma_1, \quad u_0'' \leftarrow D^2 + C \cdot (v_1 + v_1') - ((u_1'' - CC) \cdot \sigma_1 + (U_1 + U_1'))/2,$		$2\mathbf{M} + 1\mathbf{S}$
	$U_1'' \leftarrow u_1''^2, \quad U_0'' \leftarrow u_1'' \cdot u_0'', v_1'' \leftarrow D \cdot (u_1 - u_1'') + U_1'' - u_0'' - U_1 + u_0,$		$2\mathbf{M} + 1\mathbf{S}$
	$v_0'' \leftarrow D \cdot (u_0 - u_0'') + U_0'' - U_0, \quad v_1'' \leftarrow E \cdot v_1'' + v_1 \quad v_0'' \leftarrow E \cdot v_0'' + v_0.$		3M
Output:	$D'' = \rho(D \oplus D') = (u_1'', u_0'', v_1'', v_0'', U_1'' = u_1''^2, U_0'' = u_1'' u_0'').$	**Total**	$\mathbf{I} + 17\mathbf{M} + 4\mathbf{S}$
PROJECTIVE ADDITION			
Input:	$D = (U_1, U_0, V_1, V_0, Z)$, $D' = (U_1', U_0', V_1', V_0', Z')$,		Operations
	$ZZ \leftarrow Z_1 \cdot Z_2, \quad U1Z \leftarrow U_1 \cdot Z_2, \quad U1Z' \leftarrow U_1' \cdot Z_1, \quad U1ZS \leftarrow U1Z^2, \quad U1ZS' \leftarrow U1Z'^2,$		$3\mathbf{M} + 2\mathbf{S}$
	$U0Z \leftarrow U_0 \cdot Z_2, \quad U0Z' \leftarrow U_0' \cdot Z_1, \quad V1Z \leftarrow V_1 \cdot Z_2, \quad V1Z' \leftarrow V_1' \cdot Z_1,$		4M
	$M_1 \leftarrow U1ZS - U1ZS' + ZZ \cdot (U0dZ - U0Z), \quad M_2 \leftarrow U1Z' \cdot U0Z' - U1Z \cdot U0Z;$		3M
	$M_3 \leftarrow U1Z - U1Z', \quad M_4 \leftarrow U0Z' - U0Z, \quad z_1 \leftarrow V0 \cdot Z_2 - V0' \cdot Z_1, \quad z_2 \leftarrow V1Z - V1Z',$		2M
	$t_1 \leftarrow (M_2 - z_1) \cdot (z_2 - M_1), \quad t_2 \leftarrow (-z_1 - M_2) \cdot (z_2 + M_1),$		2M
	$t_3 \leftarrow (-z_1 + M_4) \cdot (z_2 - M_3), \quad t_4 \leftarrow (-z_1 - M_4) \cdot (z_2 + M_3),$		2M
	$\ell_2 \leftarrow t_1 - t_2, \quad \ell_3 \leftarrow t_3 - t_4, \quad d \leftarrow t_3 + t_4 - t_1 - t_2 - 2 \cdot (M_2 - M_4) \cdot (M_1 + M_3),$		1M
	$A \leftarrow d^2, \quad B \leftarrow \ell_3 \cdot ZZ, \quad C \leftarrow \ell_2 \cdot B, \quad D \leftarrow d \cdot B, \quad E \leftarrow \ell_3 \cdot B, \quad F \leftarrow U1Z \cdot E, \quad G \leftarrow ZZ \cdot E,$		$6\mathbf{M} + 1\mathbf{S}$
	$H \leftarrow U0Z \cdot G, \quad J \leftarrow D \cdot G, \quad K \leftarrow Z_2 \cdot J, \quad U_1'' \leftarrow 2 \cdot C - A - E \cdot (U1Z + U1Z'),$		4M
	$U_0'' \leftarrow \ell_2^2 \cdot ZZ + D \cdot (V1Z + V1Z') - ((U_1'' - A) \cdot (U1Z + U1Z') + E \cdot (U1ZS + U1ZS'))/2,$		$4\mathbf{M} + 1\mathbf{S}$
	$V_1'' \leftarrow U_1'' \cdot (U_1'' - C) + F \cdot (C - F) + E \cdot (H - U_0''),$		3M
	$V_0'' \leftarrow H \cdot (C - F) + U_0'' \cdot (U_1'' - C), \quad V_1'' \leftarrow V_1'' \cdot ZZ + K \cdot V_1, \quad V_0'' \leftarrow V_0'' + K \cdot V_0,$		5M
	$U_1'' \leftarrow U_1'' \cdot D \cdot ZZ, \quad U_0'' \leftarrow U_0'' \cdot D, \quad Z'' \leftarrow ZZ \cdot J.$		4M
Output:	$D'' = \rho(D \oplus D') = (U_1'', U_0'', V_1'', V_0'', Z'').$	**Total**	$43\mathbf{M} + 4\mathbf{S}$
AFFINE DOUBLING			
Input:	$D = (u_1, u_0, v_1, v_0, U_1 = u_1^2, U_0 = u_1 u_0)$, with constants f_2, f_3		Operations
	$vv \leftarrow v_1^2, \quad vu \leftarrow (v_1 + u_1)^2 - vv - U_1, \quad M_1 \leftarrow 2v_0 - 2vu, \quad M_2 \leftarrow 2v_1 \cdot (u_0 + 2U_1),$		$1\mathbf{M} + 2\mathbf{S}$
	$M_3 \leftarrow -2v_1, \quad M_4 \leftarrow vu + 2v_0, \quad z_1 \leftarrow f_2 + 2U_1 \cdot u_1 + 2U_0 - vv, \quad z_2 \leftarrow f_3 - 2u_0 + 3U_1,$		1M
	$t_1 \leftarrow (M_2 - z_1) \cdot (z_2 - M_1), \quad t_2 \leftarrow (-z_1 - M_2) \cdot (z_2 + M_1),$		2M
	$t_3 \leftarrow (M_4 - z_1) \cdot (z_2 - M_3), \quad t_4 \leftarrow (-z_1 - M_4) \cdot (z_2 + M_3),$		2M
	$\ell_2 \leftarrow t_1 - t_2, \quad \ell_3 \leftarrow t_3 - t_4, \quad d \leftarrow t_3 + t_4 - t_1 - t_2 - 2(M_2 - M_4) \cdot (M_1 + M_3),$		1M
	$A \leftarrow 1/(d \cdot \ell_3), \quad B \leftarrow d \cdot A, \quad C \leftarrow d \cdot B, \quad D \leftarrow \ell_2 \cdot B, \quad E \leftarrow \ell_3^2 \cdot A,$		$\mathbf{I} + 5\mathbf{M} + 1\mathbf{S}$
	$u_1'' \leftarrow 2D - C^2 - 2u_1, \quad u_0'' \leftarrow (D - u_1)^2 + 2C \cdot (v_1 + C \cdot u_1), \quad U_1'' \leftarrow u_1''^2, \quad U_0'' \leftarrow u_1'' \cdot u_0'',$		$3\mathbf{M} + 3\mathbf{S}$
	$v_1'' \leftarrow D \cdot (u_1 - u_1'') + U_1'' - U_1 - u_0'' + u_0, \quad v_0'' \leftarrow D \cdot (u_0 - u_0'') + U_0'' - U_0,$		2M
	$v_1'' \leftarrow E \cdot v_1'' + v_1, \quad v_0'' \leftarrow E \cdot v_0'' + v_0.$		2M
Output:	$D'' = \rho([2]D) = (u_1'', u_0'', v_1'', v_0'', U_1'' = u_1''^2, U_0'' = u_1'' u_0'').$	**Total**	$\mathbf{I} + 19\mathbf{M} + 6\mathbf{S}$
PROJECTIVE DOUBLING			
Input:	$D = (U_1, U_0, V_1, V_0, Z)$, curve constants f_2, f_3		Operations
	$UU \leftarrow U_1 \cdot U_0, \quad U_{1S} \leftarrow U_1^2, \quad Z_S \leftarrow Z^2, \quad V0Z \leftarrow V0 \cdot Z, \quad U0Z \leftarrow U0 \cdot Z, \quad V_{1S} \leftarrow V1^2,$		$3\mathbf{M} + 3\mathbf{S}$
	$UV \leftarrow (V_1 + U_1)^2 - V_{1S} - U_{1S}, \quad M_1 \leftarrow 2 \cdot V0Z - 2 \cdot UV, \quad M_2 \leftarrow 2 \cdot V1 \cdot (U0Z + 2 \cdot U_{1S}),$		$1\mathbf{M} + 1\mathbf{S}$
	$M_3 \leftarrow -2 \cdot V_1, \quad M_4 \leftarrow UV + 2 \cdot V0Z, \quad z_1 \leftarrow Z \cdot (f_2 \cdot Z_S - V_{1S}) + 2 \cdot U_1 \cdot (U_{1S} + U0Z),$		2M
	$z_2 \leftarrow f_3 \cdot Z_S - 2 \cdot U0Z + 3 \cdot U_{1S}, \quad t_1 \leftarrow (M_2 - z1) \cdot (z_2 - M_1), \quad t_2 \leftarrow (-z_1 - M_2) \cdot (z_2 + M_1),$		2M
	$t_3 \leftarrow (-z_1 + M_4) \cdot (z_2 - M_3), \quad t_4 \leftarrow (-z_1 - M_4) \cdot (z_2 + M_3),$		2M
	$\ell_2 \leftarrow t_1 - t_2, \quad \ell_3 \leftarrow t_3 - t_4, \quad d \leftarrow t_3 + t_4 - t_1 - t_2 - 2 \cdot (M_2 - M_4) \cdot (M_1 + M_3),$		1M
	$A \leftarrow \ell_2^2, \quad B \leftarrow \ell_3^2, \quad C \leftarrow ((\ell_2 + \ell_3)^2 - A - B)/2, \quad D \leftarrow B \cdot Z, \quad E \leftarrow B \cdot U_1,$		$2\mathbf{M} + 3\mathbf{S}$
	$F \leftarrow d^2, \quad G \leftarrow F \cdot Z, \quad H \leftarrow ((d + \ell_3)^2 - F - B)/2, \quad J \leftarrow H \cdot Z, \quad K \leftarrow V_1 \cdot J, \quad L \leftarrow U0Z \cdot B,$		$4\mathbf{M} + 2\mathbf{S}$
	$U_1'' \leftarrow 2 \cdot C - 2 \cdot E - G, \quad U_0'' \leftarrow A + U_1 \cdot (E - 2 \cdot C + 2 \cdot G) + 2 \cdot K,$		1M
	$V_1'' \leftarrow (C - E - U_1'') \cdot (E - U_1'') + B \cdot (L - U_0''), \quad V_0'' \leftarrow L \cdot (C - E) + (U_1'' - C) \cdot U_0''.$		4M
	$V_1'' \leftarrow V_1'' \cdot Z + K \cdot D, \quad V_0'' \leftarrow V_0'' + V0Z \cdot H \cdot D, \quad M \leftarrow J \cdot Z, \quad U_1'' \leftarrow U_1'' \cdot M, \quad U_0'' \leftarrow U_0'' \cdot J,$		7M
	$Z'' \leftarrow M \cdot D.$		1M
Output:	$D'' = \rho([2]D) = (U_1'', U_0'', V_1'', V_0'', Z'').$	**Total**	$30\mathbf{M} + 9\mathbf{S}$

Table 7.1: Explicit formulas for a divisor addition $D'' = D \oplus D'$ involving two distinct degree 2 divisors on $\mathrm{Jac}(C_2)$, and for divisor doubling $D'' = [2]D$ of a degree 2 divisor on $\mathrm{Jac}(C_2)$.

u_1^2 and u_1u_0.

7.4.2 General divisor doubling in genus 2

Let $D = (x^2+u_1x+u_0, v_1x+v_0) \in \hat{\mathrm{J}}\mathrm{ac}(C_2)$ be a divisor with $\mathrm{supp}(D) = \{P_1, P_2\} \cup \{P_\infty\}$. To compute $[2]D = D \oplus D$, we seek the cubic polynomial $\ell(x) = \sum_{i=0}^{3} \ell_i x^i$ that has zeroes of order two at both $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$. We can immediately make use of the equations arising out of the interpolation of $\mathrm{supp}(D)$ in (7.3) to obtain the first $g = 2$ equations.

There are two possible approaches to obtaining the second set of $g = 2$ equations. The first is the geometric flavoured approach that was used in the proof of Proposition 7.5 and in Example 7.3.3, which involves matching the derivatives. The second involves reducing the substitution of $\ell(x)$ into C_g by $\langle u(x)^2 \rangle$ to ensure the prescribed zeros are of multiplicity two, and using the associated Mumford ideals to linearise the equations. For the purpose of presenting both approaches, we will illustrate the latter approach in this subsection, but it is important to highlight that the guaranteed existence of linear equations follows from the expression gained when matching derivatives in the geometric approach.

We start by setting $y = \ell(x)$ into C_2 and reducing modulo the ideal $\langle (x^2 + u_1x + u_0)^2 \rangle$, which gives

$$\Omega(x) = \Omega_0 + \Omega_1 x + \Omega_2 x^2 + \Omega_3 x^3 \equiv (\sum_{i=0}^{3} \ell_i x^i)^2 - f(x) \mod \langle (x^2 + u_1x + u_0)^2 \rangle$$

where

$$\begin{aligned}
\Omega_0 &= \ell_3^2(2u_0^3 - 3u_1^2u_0^2) + 4\ell_3\ell_2u_1u_0^2 - 2\ell_3\ell_1u_0^2 + \ell_0^2 - \ell_2^2u_0^2 - 2u_1u_0^2 - f_0, \\
\Omega_1 &= 6\ell_3^2(u_1u_0^2 - u_1^3u_0) + 2\ell_3\ell_2(4u_1^2u_0 - u_0^2) + 2\ell_1\ell_0 - 4\ell_3\ell_1u_0u_1 \\
&\quad - 2\ell_2^2u_0u_1 - 4u_1^2u_0 + u_0^2 - f_1, \\
\Omega_2 &= 3\ell_3^2(u_0^2 - u_1^4) + \ell_1^2 - \ell_2^2(u_1^2 + 2u_0) - 2u_0u_1 - 2u_1^3 + 4\ell_3\ell_2(u_1^3 + u_0u_1) \\
&\quad - 2\ell_3\ell_1(2u_0 + u_1^2) + 2\ell_2\ell_0 - f_2, \\
\Omega_3 &= 2\ell_3^2(3u_1u_0 - 2u_1^3) + 2\ell_2\ell_1 + 2\ell_3\ell_2(3u_1^2 - 2u_0) - 2\ell_2^2u_1 - 4\ell_3\ell_1u_1 + 2\ell_3\ell_0 \\
&\quad - 3u_1^2 + 2u_0 - f_3.
\end{aligned}$$

It follows that $\Omega_i = 0$ for $0 \leq i \leq 3$. Although we now have four more equations relating the unknown ℓ_i coefficients, these equations are currently nonlinear.

We linearize by substituting the linear equations taken from (7.3) above, and reducing the results modulo the Mumford ideals given in (7.2). We use the two linear equations $\tilde{\Omega}_2, \tilde{\Omega}_3$ resulting from Ω_2, Ω_3, given as

$$\begin{aligned}\tilde{\Omega}_2 &= 4\ell_1 v_1 + 2\ell_2(v_0 - 2v_1u_1) - 6\ell_3 u_0 v_1 - 2u_0u_1 - 2u_1^3 - 3v_1^2 - f_2,\\ \tilde{\Omega}_3 &= 2v_1\ell_2 + \ell_3(2v_0 - 4u_1v_1) + 2u_0 - 3u_1^2 - f_3,\end{aligned}$$

which combine with the linear interpolating equations (in (7.3)) to give rise to the linear system

$$\begin{pmatrix} -1 & 0 & u_0 & -u_1u_0 \\ 0 & -1 & u_1 & -u_1^2 + u_0 \\ 0 & 4v_1 & 2v_0 - 2v_1u_1 & -6u_0v_1 \\ 0 & 0 & 2v_1 & -4v_1u_1 + 2v_0 \end{pmatrix} \cdot \begin{pmatrix} \ell_0 \\ \ell_1 \\ \ell_2 \\ \ell_3 \end{pmatrix} = \begin{pmatrix} -v_0 \\ -v_1 \\ f_2 + 2u_1u_0 + 2u_1^3 + 3v_1^2 \\ f_3 - 2u_0 + 3u_1^2 \end{pmatrix}.$$

As was the case with the divisor addition in the previous section, we can first solve a smaller system for ℓ_2 and ℓ_3, by adding the appropriate multiple of the second row to the third row above, to give

$$\begin{pmatrix} 2v_1u_1 + 2v_0 & -2u_0v_1 - 4v_1u_1^2 \\ 2v_1 & -4v_1u_1 + 2v_0 \end{pmatrix} \cdot \begin{pmatrix} \ell_2 \\ \ell_3 \end{pmatrix} = \begin{pmatrix} f_2 + 2u_1u_0 + 2u_1^3 - v_1^2 \\ f_3 - 2u_0 + 3u_1^2 \end{pmatrix}.$$

After solving the above system for ℓ_2 and ℓ_3, the process of obtaining $D'' = [2]D = (x^2 + u_1''x + u_0'', v_1''x + v_0'')$ is identical to the case of addition in the previous section, giving rise to the analogous explicit formulas in Table 7.1.

7.4.3 Comparisons of formulas in genus 2

Table 7.2 draws comparisons between the explicit formulas obtained from the above approach and the explicit formulas presented in previous work. In implementations where inversions are expensive compared to multiplications (i.e. $\mathbf{I} > 20\mathbf{M}$), it can be advantageous to adopt projective formulas which avoid inversions altogether. Our projective formulas compute scalar multiples faster than all previous projective formulas for general genus 2 curves. We also note that our homogeneous projective formulas require only 5 coordinates in total, which is the heuristic minimum for projective implementations in genus 2.

In the case of the affine formulas, it is worth commenting that, unlike the case of elliptic curves where point doublings are generally much faster than additions, affine genus 2 operations reveal divisor additions to be the significantly cheaper

$\mathbb{F}_q$ inversions	Previous work	#	Doubling		Addition		Mixed	
I		coords	**M**	**S**	**M**	**S**	**M**	**S**
2	Harley [Har, GH00]	4	30	-	24	3	-	
	Lange [Lan01]	4	24	6	24	3	-	
	Matsuo *et al.* [MCT01]	4	27	-	25	-	-	
1	Takahashi [Tak02]	4	29	-	25	-	-	
	Miyamoto *et al.* [MDM$^+$02]	4	27	-	26	-	-	
	Lange [Lan05]	4	22	5	22	3	-	
	This work	**6**	**19**	**6**	**17**	**4**	-	
-	Wollinger and Kovtun [WK07]	5	39	6	46	4	39	4
	Lange [Lan02b, Lan05]	5	38	6	47	4	40	3
	Fan *et al.* [FGJ08]	5	39	6	-	-	38	3
	Fan *et al.* [FGJ08]	8	35	7	-	-	36	5
	Lange [Lan02c, Lan05]	8	34	7	47	7	36	5
	This work	**5**	**30**	**9**	**43**	**4**	**36**	**5**

Table 7.2: Comparisons between our explicit formulas for genus 2 curves over prime fields and previous formulas using CRT based composition.

operation. In cases where an addition would usually follow a doubling to compute $[2]D \oplus D'$, it is likely to be computationally favourable to instead compute $(D \oplus D') \oplus D$, provided temporary storage of the additional intermediate divisor is not problematic.

Lastly, the formulas in Table 7.1 all required the solution to a linear system of dimension 2. This would ordinarily require 6 $\mathbb{F}_q$ multiplications, but we applied Hisil's trick [His10, eq. 3.8] to instead perform these computations using 5 $\mathbb{F}_q$ multiplications. In implementations where extremely optimised multiplication routines give rise to $\mathbb{F}_q$ addition costs that are relatively high compared to $\mathbb{F}_q$ multiplications, it may be advantageous to undo such tricks (including **M**-**S** trade-offs) in favour of a lower number of additions.

7.5 The general description

This section presents the algorithm for divisor composition on hyperelliptic Jacobians of any genus g. The general method for reduction has essentially remained the same in all related publications following Cantor's original paper (at least in the case of low genera), but we give a simple geometric interpretation of the number of reduction rounds required in Section 7.5.3 below.

7.5.1 Composition for $g \geq 2$

We extend the composition described for genus 2 in sections 7.4.1 and 7.4.2 to hyperelliptic curves of arbitrary genus. Importantly, there are two aspects of this general description to highlight.

(i) In contrast to Cantor's general description of composition which involves polynomial arithmetic, this general description is immediately explicit in terms of $\mathbb{F}_q$ arithmetic.

(ii) The required function $\ell(x)$ is of degree $2g-1$ and therefore has $2g$ unknown coefficients. Thus, we would usually expect to solve a linear system of dimension $2g$, but the linear system that requires solving in the Mumford function field is actually of dimension g.

Henceforth we use $\mathbf{M} \cdot \mathbf{x} = \mathbf{z}$ to denote the associated linear system of dimension g, and we focus our discussion on the structure of $\mathbf{M}$ and $\mathbf{z}$.

In the case of a general divisor addition, $\mathbf{M}$ is computed as $\mathbf{M} = \mathbf{U} - \mathbf{U}'$, where $\mathbf{U}$ and $\mathbf{U}'$ are described by D and D' respectively. In fact, as for the system derived from coordinates of points above, the matrix $\mathbf{M}$ is completely dependent on $u(x)$ and $u'(x)$, whilst the vector $\mathbf{z}$ depends entirely on $v(x)$ and $v'(x)$. Algorithm 7.1 details how to build $\mathbf{U}$ (resp. $\mathbf{U}'$), where the first column of $\mathbf{U}$ is initialised as the Mumford coordinates $\{u_i\}_{1 \leq i < g}$ of D, and the remaining $g^2 - g$ entries are computed by proceeding across the columns and taking $\mathbf{U}_{i,j} = u_{i-1} \cdot \mathbf{U}_{g,j-1} + \mathbf{U}_{i-1,j-1}$. This relationship is obtained by a careful generalisation of the process that computed (7.4) from (7.3) in the case of genus 2.

Depending on the genus, we remark that Algorithm 7.1 will most likely not be the fastest way to compute $\mathbf{M}$. Instead, we propose that a faster routine is likely to be achieved by using Algorithm 7.1 to determine the algebraic expression for each of the elements in $\mathbf{M}$, and tailor making optimized formulas to generate its entries, in the same way that the previous section did for genus 2.

In addition, there is alternative way to view the structure (and computation) of the matrix $\mathbf{M}$. This follows from observing that both $\mathbf{U}$ and $\mathbf{U}'$ can actually be written as a sum of g matrices that are computed as outer products; let $\mathbf{c} = (c_1, .., c_g), \tilde{\mathbf{c}} = (\tilde{c}_1, ..., \tilde{c}_g) \in \mathbb{F}_q^g$ be two vectors that are derived solely from

Algorithm 7.1 General composition (addition) of two distinct divisors.

Input: $D = \{u_i, v_i\}_{0 \le i \le g-1}$, $D' = \{u'_i, v'_i\}_{0 \le i \le g-1}$.

Output: $\ell(x) = \sum_{i=0}^{2g-1} \ell_i x^i$ such that $\mathrm{supp}(D) \cup \mathrm{supp}(D') \subset \mathrm{supp}(\mathrm{div}(\ell))$.

1: $\mathbf{U}, \mathbf{U}', \mathbf{M} \leftarrow \{0\}^{g \times g} \in \mathbb{F}_q^{g \times g}$, $\mathbf{z} \leftarrow \{0\}^g \in \mathbb{F}_q^g$.
2: **for** i from 1 to g **do**
3: $\quad \mathbf{U}_{g+1-i,1} \leftarrow -u_{g-i} \quad ; \quad \mathbf{U}'_{g+1-i,1} \leftarrow -u'_{g-i}$
4: **end for**
5: **for** j from 2 to g **do**
6: $\quad \mathbf{U}_{1,j} \leftarrow \mathbf{U}_{g,j-1} \cdot \mathbf{U}_{1,1} \quad ; \quad \mathbf{U}'_{1,j} \leftarrow \mathbf{U}'_{g,j-1} \cdot \mathbf{U}'_{1,1}$.
7: $\quad$ **for** i from 2 to g **do**
8: $\quad\quad \mathbf{U}_{i,j} \leftarrow \mathbf{U}_{g,j-1} \cdot \mathbf{U}_{i,1} + \mathbf{U}_{i-1,j-1} \quad ; \quad \mathbf{U}'_{i,j} \leftarrow \mathbf{U}'_{g,j-1} \cdot \mathbf{U}'_{i,1} + \mathbf{U}'_{i-1,j-1}$.
9: $\quad$ **end for**
10: **end for**
11: $\mathbf{M} \leftarrow \mathbf{U} - \mathbf{U}'$.
12: **for** i from 1 to g **do**
13: $\quad \mathbf{z}_i \leftarrow v_{i-1} - v'_{i-1}$
14: **end for**
15: Solve $\mathbf{M} \cdot \mathbf{x} = \mathbf{z}$
16: Compute $\tilde{\mathbf{x}} = \mathbf{U} \cdot \mathbf{x}$
17: **for** i from 1 to g **do**
18: $\quad \tilde{\mathbf{x}}_i \leftarrow v_{g-i} - \tilde{\mathbf{x}}_i$
19: **end for**
20: **return** $\ell(x)$ $\quad$ (from $\tilde{\mathbf{x}} = \{\ell_0, ..., \ell_{g-1}\}$ and $\mathbf{x} = \{\ell_g, ..., \ell_{2g-1}\}$)

the g Mumford coordinates belonging to D, then $\mathbf{U}$ is given by the sum

$$\begin{pmatrix} c_1\tilde{c}_1 & \dots & c_1\tilde{c}_g \\ c_2\tilde{c}_1 & \dots & c_2\tilde{c}_g \\ \vdots & \ddots & \vdots \\ c_{g-1}\tilde{c}_1 & \dots & c_{g-1}\tilde{c}_g \\ c_g\tilde{c}_1 & \dots & c_g\tilde{c}_g \end{pmatrix} + \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & c_1\tilde{c}_1 & \dots & c_1\tilde{c}_{g-1} \\ \vdots & \dots & \ddots & \vdots \\ 0 & c_{g-2}\tilde{c}_2 & \dots & c_{g-2}\tilde{c}_{g-1} \\ 0 & c_{g-1}\tilde{c}_2 & \dots & c_{g-1}\tilde{c}_{g-1} \end{pmatrix} + \dots + \begin{pmatrix} 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & \dots & 0 & 0 \\ 0 & \dots & 0 & c_1\tilde{c}_1 \end{pmatrix}.$$

Example 7.5.1. Assume a general genus 3 curve and let the Mumford representations of the divisors D and D' be as usual. The matrix $\mathbf{U}$ is given as

$$\mathbf{U} = \begin{pmatrix} -u_0 & u_2u_0 & (-u_2^2+u_1)u_0 \\ -u_1 & u_2u_1 & (-u_2^2+u_1)u_1 \\ -u_2 & u_2^2 & (-u_2^2+u_1)u_2 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & -u_0 & u_2u_0 \\ 0 & -u_1 & u_2u_1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -u_0 \end{pmatrix},$$

and $\mathbf{U}'$ is given identically. In this case $\mathbf{c} = (u_0, u_1, u_2)^T$ and $\tilde{\mathbf{c}} = (-1, u_2, -u_2^2 + u_1)^T$. Setting $\mathbf{M} = \mathbf{U} - \mathbf{U}'$ and $\mathbf{z} = (v_0 - v'_0, v_1 - v'_1, v_2 - v'_2)^T$, we find the $g = 3$ coefficients ℓ_3, ℓ_4 and ℓ_5 of the quintic $\ell(x) = \sum_{i=0}^{5} \ell_i x^i$ that interpolates the 6 non-trivial elements in $\mathrm{supp}(D) \cup \mathrm{supp}(D')$ by solving $\mathbf{M} \cdot \mathbf{x} = \mathbf{z}$ for

$\mathbf{x} = (\ell_3, \ell_4, \ell_5)^T$. The remaining coefficients are found via a straightforward matrix multiplication as $\tilde{\mathbf{x}} = (\ell_0, \ell_1, \ell_2)^T = \mathbf{U} \cdot \mathbf{x}$.

The immediate observation in general is that $\mathbf{c}\tilde{\mathbf{c}}^T$ is the only outer product that requires computation in order to determine $\mathbf{U}$ entirely.

For general divisor doublings the description of the linear system is much longer; this is because the right hand side vector $\mathbf{z}$ is slightly more complicated than in the case of addition: as is the case with general Weierstrass elliptic curves, additions tend to be independent of the curve constants whilst doublings do not. We reiterate that, for low genus implementations at least, Algorithm 7.2 is intended to obtain the algebraic expressions for each element in $\mathbf{M}$; as was the case with genus 2, a faster computational route to determining the composition function will probably arise from genus specific attention that derives tailor-made explicit formulas. Besides, the general consequence of Remark 7.4.2 is that many (if not all) of the values constituting $\mathbf{U}$ will have already been computed in the previous point operation, and can therefore be temporarily stored and reused.

7.5.2 Handling special cases

The description of divisor composition herein naturally encompasses the special cases where either (or both) of the divisors have degree less than g. In fact, Proposition 7.1 trivially generalises to describe the set of divisors on $\mathrm{Jac}(C_g)$ whose effective parts have degree $d \leq g$, and can therefore be used to obtain the Mumford ideals associated with special input divisors[3].

This will often result in fewer rounds of reduction and a simpler linear system. For example, whilst the general addition of two full degree divisors in genus 3 requires an additional round of reduction after the first points of intersection are found (see Figure 1 and Figure 2), it is easy to see that any group operation on a genus 3 curve involving a divisor of degree less than 3 will give rise to a reduced divisor immediately. Clearly, the linear systems in these cases are smaller, and therefore the explicit formulas arising in these special cases will always be much faster, in agreement with all prior expositions (cf. [ACD+05, §14]). In higher genus implementations that do not explicitly account for all special cases of inputs, Katagi *et al.* [KKAT04] noted that it can still be very advantageous to

[3]Perhaps the most general consequence of Proposition 7.1 is using it to describe (or enumerate) the entire Jacobian by summing over all d, as $\#\mathrm{Jac}(C_g) = \#C_g + \sum_{d=2}^{g} n_d$, where n_d is the number of $2d$-tuples lying in the intersection of the d associated hypersurfaces.

Algorithm 7.2 General composition (doubling) of a unique divisor with itself.

Input: $D = \{u_i, v_i\}_{0 \le i \le g-1}$ and curve coefficients $f_0, f_1, ..., f_{2g-1}$.

Output: $\ell(x) = \sum_{i=0}^{2g-1} \ell_i x^i$ such that each non-trivial element in supp(D) occurs with multiplicity two in div(ℓ) .

$\mathbf{U}, \mathbf{M} \leftarrow \{0\}^{g \times g} \in \mathbb{F}_q^{g \times g}$, $\mathbf{v} \leftarrow \{0\}^{g-1} \in \mathbb{F}_q^{g-1}$, $\mathbf{z} \leftarrow \{0\}^{g} \in \mathbb{F}_q^{g}$
for i from 1 to g **do**
    $\mathbf{U}_{g+1-i,1} \leftarrow -u_{g-i}$
end for
for j from 2 to g **do**
    $\mathbf{U}_{1,j} \leftarrow \mathbf{U}_{g,j-1} \cdot \mathbf{U}_{1,1}$.
    for i from 2 to g **do**
        $\mathbf{U}_{i,j} \leftarrow \mathbf{U}_{g,j-1} \cdot \mathbf{U}_{i,1} + \mathbf{U}_{i-1,j-1}$.
    end for
end for
$u_{\text{extra}} \leftarrow \mathbf{U}_{g,1} \cdot \mathbf{U}_{g,g} + \mathbf{U}_{g-1,g}$.
for i from 1 to g **do**
    $\mathbf{M}_{g+1-i,1} \leftarrow v_{g-i}$
end for
for j from 2 to g **do**
    $\mathbf{M}_{i,j} \leftarrow \mathbf{M}_{i,j} + \mathbf{U}_{g,j-1} \cdot \mathbf{M}_{i,1} + \mathbf{M}_{g,j-1} \cdot \mathbf{U}_{i,1} + \mathbf{M}_{i-1,j-1}$.
end for
for i from 1 to $g-1$ **do**
    $\mathbf{z}_{g+1-i} \leftarrow \mathbf{z}_{g+1-i} + 2 \cdot \mathbf{U}_{g,1} \cdot \mathbf{U}_{g+1-i,1} + \mathbf{U}_{g-i,1} + \mathbf{U}_{g,i+1} + f_{2g-i}$.
    for j from 1 to i **do**
        $\mathbf{z}_{g-i} \leftarrow \mathbf{z}_{g-i} + f_{2g-1-i+j} \cdot U_{g,j}$.
        $\mathbf{v}_i \leftarrow \mathbf{v}_i - \mathbf{M}_{g+1-j,1} \cdot \mathbf{M}_{g-i+j,1}$.
    end for
end for
$\mathbf{z}_1 \leftarrow \mathbf{z}_1 + 2 \cdot \mathbf{U}_{g,1} \cdot \mathbf{U}_{1,1} + f_g$.
$\mathbf{z}_{g-1} \leftarrow \mathbf{z}_{g-1} + \mathbf{v}_1$.
for i from 3 to g **do**
    for j from 2 to $i-1$ **do**
        $\mathbf{z}_{g+1-i} \leftarrow \mathbf{z}_{g+1-i} + \mathbf{v}_{i-j} \cdot \mathbf{U}_{g,j-1}$.
    end for
    $\mathbf{z}_{g+1-i} \leftarrow \mathbf{z}_{g+1-i} + \mathbf{v}_{i-1}$.
end for
$\mathbf{z}_1 \leftarrow \mathbf{z}_1 + u_{\text{extra}}$.
for i from 1 to g **do**
    $\mathbf{z}_i \leftarrow \mathbf{z}_i / 2$.
end for
Solve $\mathbf{M} \cdot \mathbf{x} = \mathbf{z}$
Compute $\tilde{\mathbf{x}} = -\mathbf{U} \cdot \mathbf{x}$
for i from 1 to g **do**
    $\tilde{\mathbf{x}}_i \leftarrow v_{g-i} + \tilde{\mathbf{x}}_i$
end for
return $\ell(x)$ (from $\tilde{\mathbf{x}} = \{\ell_0, ..., \ell_{g-1}\}$ and $\mathbf{x} = \{\ell_g, ..., \ell_{2g-1}\}$)

explicitly implement and optimize *one* of the special cases.

7.5.3 Reduction in low genera

Gaudry's chapter [Gau05] gives an overview of different algorithms (and complexities) for the reduction phase. Our experiments lead us to believe that the usual method of reduction is still the most preferable for small g. In genus 2 we saw that point additions and doublings do not require more than one round of reduction, i.e. the initial interpolating function intersects C_2 in at most two more places (refer to Figure 3), immediately giving rise to the reduced divisor that is the sum. In genus $g \geq 3$ however, this is generally not the case. Namely, the initial interpolating function intersects C_g in more than g places, giving rise to an unreduced divisor that requires further reduction. We restate Cantor's complexity argument concerning the number of rounds of reduction [Can87, §4] in a geometric way in the following proposition.

Proposition 7.6. *In the addition of any two reduced divisor classes on the Jacobian of a genus g hyperelliptic curve, the number of rounds of further reduction required to form the reduced divisor is at most* $\lfloor \frac{g-1}{2} \rfloor$*, with equality occurring in the general case.*

Proof. For completeness note that addition on elliptic curves in Weierstrass form needs no reduction, so take $g \geq 2$. The composition polynomial $y = \ell(x)$ with the $2g$ prescribed zeros (including multiplicities) has degree $2g-1$. Substituting $y = \ell(x)$ into $C_g : y^2 + h(x)y = f(x)$ gives an equation of degree $\max\{2g+1, 3g-1, 2(2g-1)\} = 2(2g-1)$ in x, for which there are at most $2(2g-1) - 2g = 2g-2$ new roots. Let n_t be the maximum number of new roots after t rounds of reduction, so that $n_0 = 2g-2$. While $n_t > g$, reduction is not complete, so continue by interpolating the n_t new points with a polynomial of degree $n_t - 1$, producing at most $2(n_t-1) - n_t = n_t - 2$ new roots. It follows that $n_t = 2g - 2t - 2$, and since $t, g \in \mathbb{Z}$, the result follows. □

7.6 Further implications and potential

This section is intended to further illustrate the potential of coupling a geometric approach with linear algebra when performing arithmetic in Jacobians. It is our

hope that the suggestions in this section encourage future investigations and improvements.

We start by commenting that our algorithm can naturally be generalised to much more than standard divisor additions and doublings. Namely, given any set of divisors $D_1, ..., D_n \in C_g$ and any corresponding set of scalars $r_1, ..., r_n \in \mathbb{Z}$, we can theoretically compute $D = \sum_{i=1}^{n} [r_i]D_i$ at once, by first prescribing a function that, for each $1 \leq i \leq n$, has a zero of order r_i at each of the non-trivial points in the support of D_i. Note that if $r_i \notin \mathbb{Z}^+$, then prescribing a zero of order r_i at some point P is equivalent to prescribing a pole of order $-r_i \in \mathbb{Z}^+$ at P instead. We first return to genus 1 to show that this technique can be used to recover several results that were previously obtained by alternatively merging or overlapping consecutive elliptic curve computations (cf. [ELM03, CJLM06]).

7.6.1 Simultaneous operations on elliptic curves

In the case of genus 1, the Mumford representation of reduced divisors is trivial, i.e. if $P = (x_1, y_1)$, the Mumford representation of the associated divisor is $D_P = (x - x_1, y_1)$, and the associated Mumford ideal is (isomorphic to) the curve itself. However, we can again explore using the Mumford representation as an alternative to derivatives in order to generate the required linear systems arising from prescribing multiplicities of greater than one. In addition, when unreduced divisors in genus 1 are encountered, the Mumford representation becomes non-trivial and very necessary for efficient computations.

To double-and-add or point triple on an elliptic curve, we can prescribe a parabola $\ell(x) = \ell_2 x^2 + \ell_1 x + \ell_0 \in \mathbb{F}_q(E)$ with appropriate multiplicities in advance, as an alternative to Eisenträger *et al.*'s technique of merging two consecutive chords into a parabola [ELM03]. Depending on the specifics of an implementation, computing the parabola in this fashion offers the same potential advantage as that presented by Ciet *et al.* [CJLM06]; we avoid any intermediate computations and bypass computing $P + P'$ or $[2]P$ along the way. When tripling the point $P = (x_P, y_P) \in E$, the parabola is determined from the three equalities $\ell(x)^2 \equiv x^3 + f_1 x + f_0 \bmod \langle (x - u_0)^i \rangle$ for $1 \leq i \leq 3$, from which we take one of the coefficients that is identically zero in each of the three cases. As one example, we found projective formulas which compute triplings on curves of the form $y^2 = x^3 + f_0$ and cost $3\mathbf{M} + 10\mathbf{S}$. These are the second fastest tripling formulas reported across all curve models [BL07a], being only slightly slower

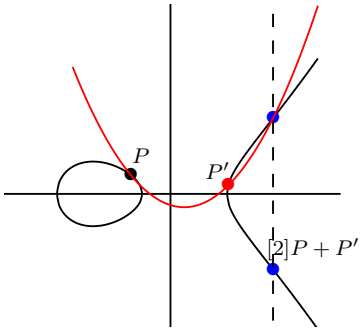


Figure 7.5: Computing $[2]P + P'$ by prescribing a parabola which intersects E at P with multiplicity three.

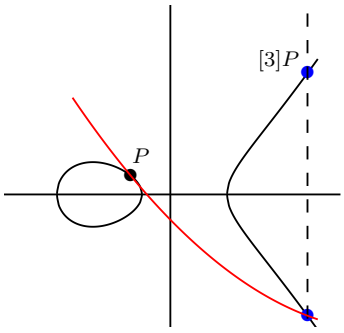


Figure 7.6: Tripling the point $P \in E$ by prescribing a parabola which intersects E at P with multiplicity three.

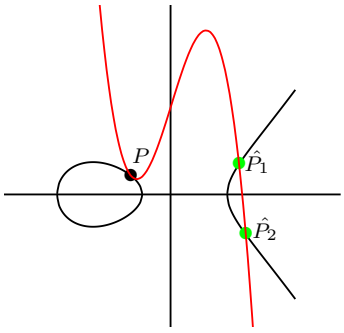


Figure 7.7: Quadrupling the point $P \in E$ by prescribing a cubic which intersects E at P with multiplicity four.

(unless $\mathbf{S} < 0.75\mathbf{M}$) than the formulas for tripling-oriented curves introduced by Doche *et al.* [DIK06] which require $6\mathbf{M} + 6\mathbf{S}$.

We can quadruple the point P by prescribing a cubic function $\ell(x) = \ell_3 x^3 + \ell_2 x^2 + \ell_1 x + \ell_0$ which intersects E at P with multiplicity four (see Figure 7.7). This time however, the cubic is zero on E in two other places, resulting in an unreduced divisor $D_{\hat{P}} = \hat{P}_1 + \hat{P}_2$, which we can represent in Mumford coordinates as $D_{\hat{P}} = (\hat{u}(x), \hat{v}(x))$ (as if it were a reduced divisor in genus 2). Our experiments agree with prior evidence that it is unlikely that point quadruplings will outperform consecutive doublings in the preferred projective cases, although we believe that one application which could benefit from this description is pairing computations, where interpolating functions are necessary in the computations. To reduce $D_{\hat{P}}$, we need the line $y = \hat{\ell}(x)$ joining $\hat{P}_1$ with $\hat{P}_2$, which can be computed via $\hat{\ell}(x) \equiv \ell(x) \bmod \langle \hat{u}(x) \rangle$. The update to the pairing function requires both $\ell(x)$ and $\hat{\ell}(x)$, as $f_{\text{upd}} = \ell(x)/\hat{\ell}(x)$. We claim that it may be attractive to compute a quadrupling in this fashion and only update the pairing function once, rather than two doublings which update the pairing functions twice, particularly in implementations where inversions don't compare so badly against multiplications [LMN10]. It is also worth pointing out that in a quadruple-and-add computation, the unreduced divisor $D_{\hat{P}}$ need not be reduced before adding an additional point P'. Rather, it could be advantageous to immediately interpolate $\hat{P}_1$, $\hat{P}_2$ and P'

with a parabola instead.

7.6.2 Simultaneous operations in higher genus Jacobians

Increasing the prescribed multiplicity of a divisor not only increases the degree of the associated interpolating function (and hence the linear system), but also generally increases the number of rounds of reduction required after composition. In the case of genus 1, we can get away with prescribing an extra zero (double-and-add or point tripling) without having to encounter any further reduction, but for genus $g \geq 2$, this will not be the case in general. For example, even when attempting to simultaneously compute $[2]D + D'$ for two general divisors $D, D' \in \mathrm{Jac}(C_2)$, the degree of the interpolating polynomial becomes 5, instead of 3, and the dimension of the linear system that arises can only be trivially reduced from 6 to 4. Our preliminary experiments seem to suggest that unless the linear system can be reduced further, it is likely that computing $[2]D + D'$ simultaneously using our technique won't be as fast as computing two consecutive straightforward operations. However, as in the previous paragraph, we argue that such a trade-off may again become favourable in pairing computations where computing the higher-degree interpolating function would save a costly function update.

7.6.3 Explicit formulas in genus 3 and 4

Developing explicit formulas for hyperelliptic curves of genus 3 and 4 has also received some attention [Wol04, WPP05, GMA+05]. It will be interesting to see if the composition technique herein can further improve these results. In light of Remark 7.4.2 and the general description in Section 7.5, the new entries in the matrix $\mathbf{M}$ will often have been already computed in the previous point operation, suggesting an obvious extension of the coordinates if the storage space permits it. Therefore the complexity of our proposed composition essentially boils down to the complexity of solving the dimension g linear system in $\mathbb{F}_q$, and so it would also be interesting to determine for which (practically useful) genera one can find tailor-made methods of solving the special linear system that arises in Section 7.5.1.

7.6.4 Characteristic two, special cases, and more coordinates

Although the proofs in Section 7.3 were for arbitrary hyperelliptic curves over general fields, Section 7.4 simplified the exposition by focusing only on finite fields of large prime characteristic. Of course, it is possible that the description herein can be tweaked to also improve explicit formulas in the cases of special characteristic two curves (see [ACD+05, §14.5]). In addition, it is possible that the geometrically inspired derivation of explicit formulas for special cases of inputs will enhance implementations which make use of these (refer to Section 7.5.2). Finally, we only employed straightforward homogeneous coordinates to obtain the projective versions of our formulas. As was the case with the previous formulas based on Cantor's composition, it is possible that extending the projective coordinate system will give rise to even faster formulas.

7.7 Summary of contributions

This chapter presents a new and explicit method of divisor composition for hyperelliptic curves. The method is based on using simple linear algebra to derive the required geometric functions directly from the Mumford coordinates of Jacobian elements. In contrast to Cantor's composition which operates in the polynomial ring $\mathbb{F}_q[x]$, the algorithm we propose is immediately explicit in terms of $\mathbb{F}_q$ operations. We showed that this achieves the current fastest general group law formulas in genus 2, and pointed out several other potential improvements that could arise from our work.

Chapter 8

Conclusions and Future Work

We surveyed the broad field of pairing computation in Chapter 2, which laid the foundations for our novel contributions in Chapters 3-7. In Chapter 3 we presented the fastest explicit formulas for cryptographic pairing computations on elliptic curves, which find immediate application in practice. In Chapter 4 we gave systematic techniques for loop unrolling in the context of Miller's algorithm, and further showed that this technique gives significant improvements when it is applied to "fixed argument pairings", i.e. pairings that allow for precomputation. In Chapter 5 we targeted pairings at the 256-bit level by looking for optimal pairing-friendly curves from the BLS family with $k = 24$ – this extended the earlier work by Pereira *et al.* [PJNB11] which focussed on the 128-bit security level and $k = 12$ BN curves. In Chapter 6 we generalised the techniques from Chapter 5 and applied them to several other families of pairing-friendly curves that target all practical levels of security. Finally, in Chapter 7 we gave a new algorithm for performing divisor arithmetic in Jacobians of hyperelliptic curves of any genus. This algorithm not only finds application within the realm of pairing-based cryptography, but also within the arena of hyperelliptic curve cryptography (HECC).

For the moment at least, it seems that the pace of progress in the computation of a standalone Tate pairing (variant) on an ordinary elliptic curve has steadied. Indeed, Vercauteren [Ver10] and Hess [Hes08] showed that the Miller loop lengths are now optimal, whilst at the same time (and in part due to our work) it seems that all the avenues for savings within each iteration of Miller's algorithm

have also been thoroughly explored. In addition, very recent results suggest that finding complete families of pairing-friendly curves with ideal ρ values (like $k = 12$ BN curves) for higher security levels is unlikely [Oka12]. Such "negative" results could discourage those in the field (or even worse, newcomers) to keep searching for further improvements. However, there is still a lot that remains to be done.

While most of the effort has justifiably been concentrated towards the computation of the pairing itself, non-pairing operations that are specific to pairing-based protocols and which are often of equal importance have received less attention. For example, fast hashing and exponentiation routines in $\mathbb{G}_2$ are a necessity, and this has attracted attention in recent years [SBC+09b,FCKRH11]. Another important avenue that warrants further investigation is protocol-specific optimisations in the context of multi-pairings or pairing products [Sco11].

Following on from our work in Chapter 4, fixed argument optimisations certainly have a lot more potential. In particular, since the final exponentiation routine dilutes the relative speed ups obtained in Tate and ate-like pairings, it is natural to consider the potential of precomputation when there is no final exponentiation. We note the Weil pairing and the hyperelliptic ate pairing [GHO+07] as two possible candidates. In the former case one might further consider the potential of a fixed argument pairing in parallel (following the recent results in [AKMRH11]), whilst in the latter case one could consider a superfluous (but very fast) tailor-made final exponentiation to dispel any security doubts.

For now it appears that pairings on curves with genus $g \geq 2$ are doomed to remain slower than pairings on elliptic curves unless hyperelliptic curves with $\rho = g\frac{\log q}{\log r}$ values much closer to 1 are found. The absence of a (necessary) final exponentiation in the hyperelliptic ate pairing [GHO+07] makes the discovery of such curves highly desirable, particularly for applications at higher levels of security where the final exponentiation of the Tate and ate pairings on elliptic curves dominates the complexity of the pairing algorithm.

Of course, one cannot rule out the existence of an algorithm that computes the Tate or Weil pairings faster than Miller's algorithm. Stange [Sta07] recently presented the first such practical alternative for Tate pairing computations via elliptic nets, and although her algorithm is currently slightly slower, there could be optimisations that push it ahead of Miller's algorithm.

In the most broad sense, we can perhaps dream that other cryptographically

suitable pairings exist beyond the realm of Weil and Tate pairings on abelian varieties over finite fields, and hope that in such a case a more efficient method of computation is possible. The discovery of such a pairing would no doubt send the community in a flurry of excitement again.

Appendix A

Implementation-friendly BLS curves with $k = 24$

Here we provide four lists of implementation-friendly BLS curves at security levels where the entire BLS family is either competitive across all families, or are clearly the current outright favorite. Each list (Table A.1 through to Table A.4) corresponds to one of the four proposed subfamilies.

Each table lists curves where x_0 is very sparse in signed binary representation, meaning here that it has weight 3, 4 or 5. Working with signed binary representation for the Miller loop parameter and the powerings in the final exponentiation can be considered standard and extends the space of nice curves compared to just using plain binary representation. Nevertheless, we also included many x_0 values which have the same plain binary representation as the signed binary representation; these are the x_0 values which share the same sign for each power of 2.

All curves given have the implementation-friendly properties outlined in the previous sections. In particular, curves in Table A.1 and Table A.3 have $x_0 \equiv 7 \pmod{72}$ and $x_0 \equiv 31 \pmod{72}$ respectively, and are given by $E : y^2 = x^3 + 1$; curves in Table A.2 have $x_0 \equiv 16 \pmod{72}$ and are given by $E : y^2 = x^3 + 4$, and curves in Table A.4 have $x_0 \equiv 64 \pmod{72}$ and are given by $E : y^2 = x^3 - 2$. In all cases all parameters are uniquely defined by the short value x_0.

The curves in all four tables were found by trying all possibilities for the signed binary representation of x_0 with a fixed weight such that x_0 belongs to the right

congruence class modulo 72. In our search, we did not find any curves in the considered range of parameter sizes where x_0 is plus or minus a power of 2 (i.e. weight 1) or where it is a binomial, a sum of two such powers (i.e. weight 2). In this sense, our search indicates that weights 3, 4 and 5 are optimal for the security levels considered in this work. The even congruences ($x_0 \equiv 16, 64 \pmod{72}$) gain the slight advantage over the odd congruences ($x_0 \equiv 7, 31 \pmod{72}$), since the last bit of the binary representation of odd congruences is obviously forced to be 1. Thus, curves in the even congruence classes commonly have weights 3 and 4 whilst curves in the odd congruence classes commonly have weights 4 and 5. On the other hand, the odd congruences both give rise to curves with $b = 1$ which would make for slightly faster point operations, but (all other things being equal) one would probably achieve a faster implementation by taking the x_0 value with the lowest weight possible, since one less bit in x_0 saves over 10 full $\mathbb{F}_{q^{24}}$ multiplications per single pairing.

Restricting x_0 to sparse values only results in a certain inflexibility when adjusting the parameter sizes to exact values, for example certain multiples of word sizes on a target implementation platform. However, recent high-speed implementations of pairings at the 128-bit security level have shown that lazy reduction techniques give significant improvements in the field tower arithmetic and thus the overall pairing computation [BGDM$^+$10, AKL$^+$11]. Such techniques can be employed efficiently when the bit size of the prime characteristic q is a few bits less than a multiple of the word size, which provides a certain space for delaying reductions for field arithmetic. In Tables A.1, A.2, A.3 and A.4 we have tried to account for this (as far as possible), by including different choices of curves at each security level that have a varying gap between the prime field size and multiples of standard word sizes 32 and 64, which are also given in the table. We believe that most implementors of pairings in software will find a suitable curve at the desired security level in our tables, or else will be able to find a suitable curve themselves with similar properties.

security level	$x_0 \equiv 7 \pmod{72}$	weight	q (bits)	words for q	r (bits)	words for r	security (bits)
192	$-1-2^8+2^{38}+2^{45}$	4	449	8×64	361	12×32	181
	$-1+2^3-2^5-2^{19}+2^{46}$	5	459		368		184
	$-1-2^{11}-2^{26}-2^{35}-2^{47}$	5	469		377		189
	$-1+2^{19}-2^{24}+2^{27}-2^{48}$	5	479		384	6×64	192
	$-1-2^{11}-2^{28}-2^{35}-2^{49}$	5	489	16×32	393	13×32	197
	$-1-2^4-2^{21}-2^{50}$	4	499		401		201
	$-1+2^{11}-2^{28}-2^{51}$	4	509		409		205
	$-1-2^{22}-2^{26}-2^{36}-2^{52}$	5	519	9×64	417	7×64	209
	$-1+2^{44}+2^{51}+2^{53}$	4	532		427		214
224	$-1-2^3-2^{29}-2^{38}-2^{55}$	5	549	18×32	441	14×32	221
	$-1+2^3-2^{11}-2^{51}+2^{56}$	5	558		448		224
	$-1-2^{15}-2^{22}-2^{56}$	4	559		449	8×64	225
	$-1-2^{16}+2^{23}-2^{28}+2^{57}$	5	569		456		228
	$-1-2^{28}+2^{51}+2^{58}$	4	579	10×64	465		233
	$-1-2^{12}-2^{28}-2^{50}-2^{58}$	5	579		465		233
256	$-1+2^{15}+2^{43}-2^{61}$	4	609	20×32	488	16×32	244
	$-1+2^{49}+2^{55}+2^{62}$	4	619		497		249
	$-1-2^{19}-2^{23}-2^{26}-2^{63}$	5	629		505		253
	$-1-2^8-2^{35}-2^{61}-2^{63}$	5	632		507		254
	$-1+2^{17}-2^{54}+2^{61}-2^{64}$	5	637		511		256
	$-1+2^{35}+2^{60}-2^{64}$	4	638		512		256
	$-1+2^{10}+2^{14}-2^{18}+2^{64}$	5	639		512		256
	$-1-2^3-2^{37}-2^{50}-2^{65}$	5	649	11×64	521	9×64	261
288	$-1-2^{52}+2^{59}-2^{81}$	4	809	13×64	648	11×64	282
	$-1-2^{26}-2^{74}+2^{82}$	4	819		656		283
	$-1+2^{21}-2^{73}-2^{82}$	4	819		657		283
	$-1-2^7-2^{13}-2^{27}-2^{82}$	5	819		657		283
	$-1-2^{11}-2^{23}-2^{32}-2^{83}$	5	829		665		285
	$-1-2^{48}-2^{52}-2^{72}-2^{84}$	5	839	14×64	673	22×32	286
	$-1+2^8-2^{12}+2^{16}-2^{85}$	5	849		680		287
	$-1-2^3+2^{31}-2^{86}$	4	859		688		289
	$-1-2^{16}-2^{20}-2^{87}$	4	869	28×32	697		290
	$-1+2^{53}-2^{56}+2^{88}$	4	879		704		292
	$-1+2^{23}+2^{67}+2^{90}$	4	899	15×64	721	12×64	295
320	$-1-2^{12}-2^{93}-2^{95}-2^{107}$	5	1069	17×64	857	14×64	317
	$-1+2^{65}-2^{75}+2^{109}$	4	1089	18×64	872		319
	$-1-2^{64}-2^{100}+2^{110}$	4	1099		880		321
	$-1+2^{15}+2^{93}-2^{111}$	4	1109		888		322
	$-1-2^{13}+2^{57}-2^{112}$	4	1119		896	28×32	323

Table A.1: BLS curves with low-weight parameter $x_0 \equiv 7 \pmod{72}$ aiming at several security levels given in the first column. The columns "words for q" and "words for r" give the necessary number of 32- or 64-bit words to store the values for q and r, respectively. The last column provides the estimated actual security by the formula in [Sma10].

security level	$x_0 \equiv 16 \pmod{72}$	weight	q (bits)	words for q	r (bits)	words for r	security (bits)
192	$2^{47} + 2^{16} - 2^5$	3	469	15×32	377	12×32	188
	$2^{47} + 2^{43} + 2^{36} + 2^3$	4	470		377		188
	$2^{47} + 2^{44} - 2^{32} + 2^7$	4	471		378		189
	$-2^{47} - 2^{45} + 2^{32} + 2^{28}$	4	472		379		189
	$-2^{48} + 2^{45} + 2^{31} - 2^7$	4	477		383		191
	$2^{48} - 2^{14} - 2^{12} - 2^4$	4	479		384		192
	$-2^{50} + 2^{21} + 2^{17} - 2^{13}$	4	499	8×64	400	7×64	200
	$2^{51} - 2^{48} + 2^{46} - 2^{16}$	4	507		407		203
	$-2^{51} + 2^{47} - 2^{22} + 2^{15}$	4	508		408		204
	$-2^{51} - 2^8 - 2^6 - 2^4$	4	509		409		204
	$-2^{51} - 2^{48} + 2^{45} + 2^{39}$	4	510		410		205
224	$2^{56} - 2^{53} - 2^{31} - 2^9$	4	557	9×64	447	7×64	223
	$-2^{56} + 2^{40} - 2^{26} - 2^6$	4	559		448		224
	$2^{56} + 2^{40} - 2^{20}$	3	559		449	15×32	224
	$2^{57} + 2^{25} + 2^{18} + 2^{11}$	4	569		457		228
	$2^{57} + 2^{54} + 2^{51} + 2^{39}$	4	571		458		229
256	$2^{63} - 2^{47} + 2^{38}$	3	629	10×64	504	8×64	252
	$2^{63} + 2^{59} + 2^{45} - 2^{17}$	4	630		505		252
	$-2^{63} - 2^{60} - 2^{44} - 2^{16}$	4	631		506		253
	$-2^{64} + 2^{61} - 2^{35} + 2^3$	4	637		511		254
	$2^{64} - 2^{46} + 2^{15} + 2^9$	4	639		512		255
288	$2^{83} - 2^{78} + 2^{60} - 2^{22}$	4	828	13×64	664	11×64	284
	$-2^{83} - 2^{46} + 2^{24}$	3	829		665		285
	$2^{83} + 2^{81} + 2^{12} + 2^7$	4	832		667		285
	$-2^{86} + 2^{82} + 2^{71} - 2^{24}$	4	858	27×32	688	22×32	289
	$2^{86} + 2^{77} - 2^{54} + 2^{27}$	4	859		689		289
	$-2^{89} + 2^{86} + 2^{28} - 2^{15}$	4	887	14×64	711	12×64	293
	$2^{89} - 2^{84} - 2^{50} + 2^{10}$	4	888		712		293
	$2^{89} + 2^{33} - 2^{29} - 2^6$	3	889		713		293
320	$2^{108} + 2^{66} - 2^{42}$	3	1079	17×64	865	14×64	318
	$-2^{108} - 2^{105} + 2^{55} + 2^{11}$	4	1081		866		318
	$2^{109} - 2^{106} + 2^{71} + 2^{22}$	4	1087		871		319
	$2^{111} + 2^{70} + 2^{66}$	3	1109	35×32	889	28×32	322
	$2^{111} + 2^{109} - 2^{100} - 2^{83}$	4	1112		891		322
	$2^{111} + 2^{110} + 2^{103} + 2^{43}$	4	1115		893		322
	$-2^{112} + 2^{107} - 2^{57} + 2^{33}$	4	1118		896		323
	$2^{112} - 2^{66} + 2^{42}$	3	1119		896		323

Table A.2: BLS curves with low-weight parameter $x_0 \equiv 16 \pmod{72}$ aiming at several security levels given in the first column. The columns "words for qs" and "words for r" give the necessary number of 32- or 64-bit words to store the values for q and r, respectively. The last column provides the estimated actual security by the formula in [Sma10].

security level	$x_0 \equiv 31 \pmod{72}$	weight	q (bits)	words for q	r (bits)	words for r	security (bits)
192	$-1+2^{16}+2^{21}+2^{45}$	4	449	8×64	361	12×32	181
	$-1-2^{17}+2^{20}-2^{36}+2^{46}$	5	459		368		184
	$-1-2^{28}-2^{37}+2^{47}$	4	469		376		188
	$-1-2^{6}-2^{16}-2^{47}$	4	469		377		189
	$-1-2^{13}-2^{25}-2^{30}+2^{48}$	5	479		384	6×64	192
	$-1-2^{15}-2^{32}-2^{48}$	4	479		385	13×32	193
	$-1+2^{27}-2^{43}-2^{48}$	4	479		385		193
	$-1-2^{15}-2^{19}-2^{31}-2^{48}$	5	479		385		193
	$-1-2^{8}+2^{15}+2^{17}-2^{50}$	5	499	16×32	400		200
	$-1-2^{8}-2^{15}+2^{51}$	4	509		408		204
	$-1+2^{18}-2^{28}-2^{52}$	4	519	9×64	417	7×64	209
224	$-1-2^{37}+2^{40}-2^{43}+2^{55}$	5	549	18×32	440	14×32	220
	$-1-2^{18}+2^{29}+2^{35}-2^{56}$	5	559		448		224
	$-1+2^{14}-2^{22}+2^{57}$	4	569		456	8×64	228
	$-1+2^{17}+2^{27}-2^{57}$	4	569		456		228
	$-1-2^{8}-2^{34}-2^{50}-2^{58}$	5	579	10×64	465		233
	$-1+2^{13}-2^{30}-2^{59}$	4	589		473		237
256	$-1+2^{16}+2^{20}-2^{24}+2^{62}$	5	619	20×32	496	16×32	248
	$-1+2^{45}-2^{49}+2^{63}$	4	629		504		252
	$-1-2^{9}+2^{11}-2^{27}+2^{64}$	5	639		512		256
	$-1-2^{10}-2^{22}-2^{24}-2^{64}$	5	639		513		257
	$-1-2^{3}-2^{36}-2^{57}-2^{65}$	5	649	11×64	521	9×64	261
	$-1+2^{20}-2^{43}+2^{58}+2^{65}$	5	649		521		261
288	$-1+2^{25}+2^{49}-2^{81}$	4	809	13×64	648	11×64	282
	$-1+2^{3}-2^{74}-2^{81}$	4	809		649		282
	$-1-2^{11}+2^{57}-2^{82}$	4	819		656		283
	$-1-2^{18}-2^{39}+2^{83}$	4	829		664		285
	$-1-2^{18}-2^{39}+2^{83}$	4	829		664		285
	$-1+2^{31}-2^{77}-2^{84}$	4	839	14×64	673	22×32	286
	$-1-2^{20}+2^{71}+2^{86}$	4	859		689		289
320	$-1-2^{39}-2^{54}+2^{107}$	4	1069	17×64	856	14×64	317
	$-1+2^{3}+2^{10}+2^{18}-2^{109}$	5	1089	18×64	872		319
	$-1+2^{26}+2^{36}+2^{57}-2^{111}$	5	1109		888		322
	$-1-2^{8}-2^{24}+2^{37}+2^{111}$	5	1109		889	28×32	322

Table A.3: BLS curves with low-weight parameter $x_0 \equiv 31 \pmod{72}$ aiming at several security levels given in the first column. The columns "words for q" and "words for r" give the necessary number of 32- or 64-bit words to store the values for q and r, respectively. The last column provides the estimated actual security by the formula in [Sma10].

security level	$x_0 \equiv 64 \pmod{72}$	weight	q (bits)	words for q	r (bits)	words for r	security (bits)
192	$-2^{16} - 2^{27} + 2^{46}$	3	459	8×64	368	12×32	184
	$2^8 + 2^{12} + 2^{40} + 2^{46}$	4	459		369		185
	$-2^{14} + 2^{19} + 2^{21} - 2^{47}$	4	469		376	6×64	188
	$-2^{12} - 2^{30} - 2^{35} - 2^{48}$	4	479		385	13×32	193
	$-2^{10} - 2^{14} - 2^{17} - 2^{48}$	4	479		385		193
	$2^{15} + 2^{22} + 2^{25} + 2^{49}$	4	489	16×32	393		197
	$2^{12} - 2^{17} - 2^{31} - 2^{49}$	4	489		393		197
	$2^{31} - 2^{33} - 2^{49}$	3	489		393		197
224	$2^{10} + 2^{21} - 2^{28} + 2^{55}$	4	549	18×32	440	14×32	220
	$-2^4 - 2^{30} + 2^{32} - 2^{56}$	4	559		448		224
	$2^5 - 2^{10} + 2^{27} + 2^{56}$	4	559		449	8×64	225
	$2^8 + 2^{10} + 2^{16} - 2^{57}$	4	569		456		228
	$2^{31} + 2^{35} - 2^{41} + 2^{57}$	4	569		456		228
	$-2^7 + 2^{10} + 2^{16} + 2^{58}$	4	579	10×64	465		233
	$2^7 - 2^{25} + 2^{31} - 2^{60}$	4	599		480		240
256	$2^{19} - 2^{26} - 2^{37} - 2^{62}$	4	619	20×32	497	16×32	249
	$2^{16} - 2^{42} - 2^{60} - 2^{62}$	4	622		499		250
	$2^9 - 2^{38} - 2^{56} - 2^{63}$	4	629		505		253
	$-2^{14} + 2^{39} - 2^{56} - 2^{63}$	4	629		505		253
	$-2^{23} - 2^{42} - 2^{44} - 2^{64}$	4	639		513		257
	$-2^{12} - 2^{21} - 2^{60} - 2^{64}$	4	640		513		257
	$2^{17} - 2^{52} - 2^{54} - 2^{65}$	4	649	11×64	521	9×64	261
	$-2^{11} - 2^{35} - 2^{55} - 2^{65}$	4	649		521		261
288	$-2^{40} - 2^{49} - 2^{81}$	3	809	13×64	649	11×64	282
	$-2^7 - 2^{48} - 2^{70} - 2^{82}$	4	819		657		283
	$-2^7 - 2^{46} - 2^{54} - 2^{82}$	4	819		657		283
	$-2^{15} - 2^{17} + 2^{83}$	3	829		664		285
	$2^{41} + 2^{47} + 2^{68} + 2^{83}$	4	829		665		285
	$2^{17} + 2^{21} + 2^{29} + 2^{84}$	4	839	14×64	673	22×32	286
	$2^9 + 2^{13} + 2^{34} + 2^{85}$	4	849		681		287
	$-2^{31} - 2^{66} + 2^{86}$	3	859		688		289
	$2^8 + 2^{26} + 2^{34} + 2^{86}$	4	859		689		289
	$2^{34} - 2^{82} - 2^{87}$	3	869	28×32	697		290
	$2^{13} + 2^{17} - 2^{27} - 2^{88}$	4	879		705		292
	$-2^{10} - 2^{12} - 2^{14} - 2^{89}$	4	889		713		293
	$-2^6 - 2^{45} + 2^{90}$	3	899	15×64	720	12×64	295
	$2^{60} - 2^{67} + 2^{91}$	3	909		728		296
320	$2^{14} + 2^{17} - 2^{38} + 2^{107}$	4	1069	17×64	856	14×64	317
	$2^6 - 2^{32} + 2^{39} + 2^{107}$	4	1069		857		317
	$2^5 - 2^{18} - 2^{25} - 2^{108}$	4	1079		865		318
	$2^{20} + 2^{49} + 2^{71} + 2^{111}$	4	1109	18×64	889	28×32	322

Table A.4: BLS curves with low-weight parameter $x_0 \equiv 64 \pmod{72}$ aiming at several security levels given in the first column. The columns "words for q" and "words for r" give the necessary number of 32- or 64-bit words to store the values for q and r, respectively. The last column provides the estimated actual security by the formula in [Sma10].

Appendix B

Implementation-friendly curves for attractive families with $k \leq 50$

We provide numerous examples of low (Hamming or signed binary) weight implementation friendly curves from all of the families considered in Chapter 6. All of the curves given come from 5-star families. Those curves marked with an asterisk have Hamming weights equal to their signed binary weight - we refer back to the discussion at the beginning of Appendix A for more details on how these curves were found.

112-bit secure curves					
family	subfamily/details	x_0	weight	$\mathbb{F}_q$ (bits) / $\mathbb{F}_{q^k}$ sec.	r (bits) / $E[r]$ sec.
Brezing-Weng $k = 8$ (see §6.2)	$x \equiv 1 \bmod 16$ $T_1,\ a_1,\ D$	$1 - 2^{21} + 2^{48} - 2^{52}$	4	316 / 113	210 / 104
		$1 - 2^{46} - 2^{49} - 2^{52}$	4	318 / 113	211 / 105
		$1 + 2^{18} + 2^{52}$	3*	317 / 113	211 / 105
		$1 + 2^{12} + 2^{28} + 2^{52}$	4*	317 / 113	211 / 105
		$1 - 2^{6} + 2^{33} + 2^{53}$	4	323 / 114	215 / 107
		$1 - 2^{4} - 2^{14} - 2^{54}$	4	329 / 115	219 / 109
		$1 - 2^{24} + 2^{33} + 2^{54}$	4	329 / 115	219 / 109
		$1 - 2^{17} - 2^{35} - 2^{54}$	4	329 / 115	219 / 109
		$1 + 2^{16} - 2^{48} - 2^{54}$	4	329 / 115	219 / 109
		$1 + 2^{43} - 2^{53} + 2^{55}$	4	332 / 116	221 / 110
		$1 + 2^{9} + 2^{55}$	3*	335 / 116	223 / 111
		$1 - 2^{5} + 2^{20} - 2^{55}$	4	335 / 116	223 / 111
		$1 + 2^{18} + 2^{37} - 2^{55}$	4	335 / 116	223 / 111
		$1 - 2^{21} + 2^{40} - 2^{55}$	4	335 / 116	223 / 111
		$1 + 2^{37} + 2^{41} + 2^{55}$	4*	335 / 116	223 / 111
		$1 + 2^{5} + 2^{46} + 2^{55}$	4*	335 / 116	223 / 111
		$1 - 2^{43} - 2^{47} + 2^{55}$	4	335 / 116	223 / 111
		$1 - 2^{13} + 2^{54} - 2^{56}$	4	338 / 117	225 / 112
	$x \equiv 3 \bmod 16$ $T_1,\ a_{-2},\ D$	$-1 + 2^{2} - 2^{18} - 2^{52}$	4	317 / 113	211 / 105
		$-1 + 2^{2} + 2^{25} + 2^{27} + 2^{53}$	5	323 / 114	215 / 107
		$-1 + 2^{2} - 2^{21} - 2^{49} - 2^{53}$	5	323 / 114	215 / 107
		$-1 + 2^{2} + 2^{9} + 2^{51} - 2^{54}$	5	328 / 115	218 / 108
		$-1 + 2^{2} - 2^{4} - 2^{12} - 2^{54}$	5	329 / 115	219 / 109
		$-1 + 2^{2} - 2^{18} + 2^{20} + 2^{54}$	5	329 / 115	219 / 109
		$-1 + 2^{2} + 2^{13} + 2^{24} + 2^{54}$	5	329 / 115	219 / 109
		$-1 + 2^{2} - 2^{34} - 2^{54}$	4	329 / 115	219 / 109
		$-1 + 2^{2} - 2^{6} - 2^{18} + 2^{55}$	5	335 / 116	223 / 111
		$-1 + 2^{2} + 2^{11} - 2^{26} - 2^{55}$	5	335 / 116	223 / 111
		$-1 + 2^{2} + 2^{34} - 2^{40} + 2^{55}$	5	335 / 116	223 / 111
		$-1 + 2^{2} + 2^{11} - 2^{43} - 2^{55}$	5	335 / 116	223 / 111
		$-1 + 2^{2} + 2^{41} + 2^{48} + 2^{55}$	5	335 / 116	223 / 111
	$x \equiv 9 \bmod 16$ $T_1,\ a_1,\ M$	$1 - 2^{3} + 2^{24} + 2^{48} - 2^{53}$	5	323 / 114	214 / 106
		$1 + 2^{3} + 2^{5} - 2^{25} + 2^{53}$	5	323 / 114	215 / 107
		$1 - 2^{3} - 2^{5} + 2^{48} + 2^{53}$	5	323 / 114	215 / 107
		$1 - 2^{3} - 2^{19} - 2^{29} - 2^{54}$	5	329 / 115	219 / 109
		$1 + 2^{3} - 2^{7} - 2^{42} - 2^{54}$	5	329 / 115	219 / 109
		$1 + 2^{3} - 2^{39} + 2^{45} + 2^{54}$	5	329 / 115	219 / 109
		$1 + 2^{3} + 2^{35} - 2^{48} - 2^{54}$	5	329 / 115	219 / 109
		$1 - 2^{3} - 2^{41} - 2^{53} + 2^{55}$	5	332 / 116	221 / 110
		$1 - 2^{3} - 2^{11} + 2^{16} - 2^{55}$	5	335 / 116	223 / 111
		$1 + 2^{3} + 2^{6} + 2^{30} + 2^{55}$	5*	335 / 116	223 / 111
		$1 - 2^{3} + 2^{13} + 2^{38} - 2^{55}$	5	335 / 116	223 / 111
		$1 - 2^{3} - 2^{20} + 2^{38} - 2^{55}$	5	335 / 116	223 / 111
		$1 + 2^{3} - 2^{23} + 2^{38} + 2^{55}$	5	335 / 116	223 / 111
		$1 - 2^{3} - 2^{17} + 2^{39} + 2^{55}$	5	335 / 116	223 / 111
		$1 - 2^{3} + 2^{37} + 2^{39} - 2^{55}$	5	335 / 116	223 / 111
		$1 - 2^{3} - 2^{23} - 2^{44} + 2^{55}$	5	335 / 116	223 / 111
		$1 - 2^{3} + 2^{9} - 2^{46} + 2^{55}$	5	335 / 116	223 / 111
		$1 + 2^{3} - 2^{36} - 2^{47} - 2^{55}$	5	335 / 116	223 / 111
		$1 - 2^{3} - 2^{28} - 2^{49} - 2^{55}$	5	335 / 116	223 / 111
		$1 - 2^{3} - 2^{37} - 2^{52} - 2^{55}$	5	336 / 116	223 / 111
	$x \equiv 11 \bmod 16$ $T_1,\ a_2,\ M$	$-1 - 2^{2} - 2^{5} - 2^{13} + 2^{53}$	5	323 / 114	215 / 107
		$-1 - 2^{2} - 2^{9} + 2^{23} + 2^{53}$	5	323 / 114	215 / 107
		$-1 - 2^{2} + 2^{19} + 2^{33} + 2^{53}$	5	323 / 114	215 / 107
		$-1 - 2^{2} - 2^{12} - 2^{27} - 2^{54}$	5*	329 / 115	219 / 109
		$-1 - 2^{2} - 2^{24} - 2^{37} - 2^{54}$	5*	329 / 115	219 / 109
		$-1 - 2^{2} + 2^{37} + 2^{39} + 2^{54}$	5	329 / 115	219 / 109
		$-1 - 2^{2} - 2^{7} + 2^{13} - 2^{55}$	5	335 / 116	223 / 111
		$-1 - 2^{2} + 2^{13} + 2^{17} - 2^{55}$	5	335 / 116	223 / 111
		$-1 - 2^{2} + 2^{42} - 2^{45} + 2^{55}$	5	335 / 116	223 / 111
		$-1 - 2^{2} - 2^{25} + 2^{46} + 2^{55}$	5	335 / 116	223 / 111
		$-1 - 2^{2} - 2^{11} + 2^{48} + 2^{55}$	5	335 / 116	223 / 111
		$-1 - 2^{2} + 2^{32} + 2^{54} - 2^{56}$	5	338 / 117	225 / 112

Table B.1: Low weight curves offering 112-bit security.

192-bit secure curves					
family	subfamily/details	x_0	weight	$\mathbb{F}_q$ (bits) / $\mathbb{F}_{q^k}$ sec.	r (bits) / $E[r]$ sec.
BLS $k = 12$ (see §6.3)	$x \equiv 64 \bmod 72$ $T_1,\ b_{-2},\ D$	$2^{48} - 2^{72} - 2^{105}$	3	629 / 187	421 / 210
		$2^{23} + 2^{34} + 2^{106}$	3*	635 / 188	425 / 212
		$2^{46} + 2^{74} - 2^{108}$	3	647 / 189	432 / 215
		$-2^{71} + 2^{81} - 2^{109}$	3	653 / 190	436 / 217
		$-2^{21} + 2^{91} - 2^{109}$	3	653 / 190	436 / 217
		$-2^{49} + 2^{73} - 2^{111}$	3	665 / 192	444 / 221
		$-2^{40} - 2^{67} - 2^{111}$	3*	665 / 192	445 / 222
		$2^{79} - 2^{91} - 2^{111}$	3	665 / 192	445 / 222
	$x \equiv 16, 88 \bmod 216$ $T_1,\ b_4,\ M$	$-2^{52} + 2^{62} - 2^{105}$	3	629 / 187	420 / 209
		$2^{19} + 2^{84} - 2^{107}$	3	641 / 189	428 / 213
		$-2^{23} + 2^{96} + 2^{109}$	3	653 / 190	437 / 218
		$2^{11} - 2^{25} + 2^{110}$	3	659 / 191	440 / 219
		$-2^{41} + 2^{82} - 2^{110}$	3	659 / 191	440 / 219
		$2^{60} - 2^{107} - 2^{112}$	3	671 / 192	449 / 224
	$x \equiv 160 \bmod 216$ $T_1,\ M,\ b_{-3}$	$-2^{24} - 2^{32} - 2^{34} - 2^{105}$	4*	629 / 187	421 / 210
		$-2^{14} - 2^{16} - 2^{44} - 2^{107}$	4*	641 / 189	429 / 214
		$-2^{4} - 2^{30} - 2^{61} - 2^{108}$	4*	647 / 189	433 / 216
		$-2^{8} - 2^{45} - 2^{94} - 2^{108}$	4*	647 / 189	433 / 216
		$-2^{34} - 2^{96} - 2^{103} - 2^{108}$	4*	647 / 189	433 / 216
		$2^{15} + 2^{25} + 2^{44} + 2^{109}$	4*	653 / 190	437 / 218
		$-2^{9} - 2^{91} - 2^{99} - 2^{109}$	4*	653 / 190	437 / 218
		$-2^{16} - 2^{81} + 2^{110}$	3	659 / 191	440 / 219
		$-2^{4} - 2^{62} - 2^{101} - 2^{110}$	4*	659 / 191	441 / 220
		$2^{5} + 2^{47} + 2^{58} + 2^{111}$	4*	665 / 192	445 / 222
		$2^{23} + 2^{25} + 2^{66} + 2^{111}$	4*	665 / 192	445 / 222
		$-2^{73} - 2^{85} - 2^{93} - 2^{111}$	4*	665 / 192	445 / 222
KSS $k = 16$ (see §6.4)	$x' \equiv 61, 93 \bmod 112$ $T_1,\ a_1,\ M$	$1 + 2^{12} + 2^{25} + 2^{45} - 2^{48}$	5	469 / 186	367 / 183
		$1 - 2^{26} - 2^{33} + 2^{40} - 2^{48}$	5	471 / 187	369 / 184
		$1 + 2^{21} - 2^{39} + 2^{46} + 2^{48}$	5	474 / 187	371 / 185
		$1 - 2^{12} - 2^{42} + 2^{44} - 2^{46} + 2^{49}$	6	479 / 188	375 / 187
		$1 - 2^{5} + 2^{7} + 2^{29} + 2^{43} - 2^{49}$	6	480 / 188	376 / 187
		$1 + 2^{14} + 2^{21} + 2^{25} + 2^{30} + 2^{49}$	6*	481 / 189	377 / 188
		$1 - 2^{14} + 2^{24} + 2^{36} + 2^{46} + 2^{49}$	6	482 / 189	378 / 188
		$1 - 2^{29} + 2^{31} - 2^{41} - 2^{47} - 2^{49}$	6	484 / 189	379 / 189
		$1 - 2^{29} - 2^{36} + 2^{38} - 2^{48} + 2^{50}$	6	486 / 189	381 / 190
		$1 - 2^{20} + 2^{23} - 2^{27} + 2^{30} - 2^{50}$	6	491 / 190	385 / 192
	$x' \equiv 23, 103 \bmod 112$ $T_1,\ M,\ a_{-2}$	$-1 + 2^{2} - 2^{17} - 2^{32} - 2^{45} + 2^{49}$	6	480 / 188	376 / 187
		$-1 + 2^{2} - 2^{7} - 2^{11} + 2^{20} + 2^{49}$	6	481 / 189	377 / 188
		$-1 + 2^{2} + 2^{8} + 2^{20} + 2^{23} + 2^{49}$	6	481 / 189	377 / 188
		$-1 + 2^{2} - 2^{21} - 2^{29} + 2^{38} + 2^{49}$	6	481 / 189	377 / 188
		$-1 + 2^{2} + 2^{34} + 2^{36} + 2^{48} - 2^{50}$	6	486 / 189	381 / 190
		$-1 + 2^{2} - 2^{20} - 2^{22} + 2^{31} + 2^{50}$	6	491 / 190	385 / 192
		$-1 + 2^{2} - 2^{7} + 2^{37} + 2^{51}$	5	501 / 192	393 / 196
	$x' \equiv 5, 37 \bmod 112$ $T_1,\ a_1,\ D$	$1 + 2^{3} - 2^{17} + 2^{29} + 2^{47} - 2^{49}$	6	476 / 188	373 / 186
		$1 - 2^{3} + 2^{15} + 2^{20} + 2^{32} + 2^{49}$	6	481 / 189	377 / 188
		$1 + 2^{3} + 2^{9} - 2^{15} + 2^{38} - 2^{49}$	6	481 / 189	377 / 188
		$1 - 2^{3} + 2^{18} - 2^{32} + 2^{41} + 2^{49}$	6	481 / 189	377 / 188
		$1 - 2^{3} + 2^{30} + 2^{39} - 2^{47} - 2^{49}$	6	484 / 189	379 / 189
		$1 - 2^{3} - 2^{10} - 2^{12} + 2^{31} + 2^{50}$	6	491 / 190	385 / 192
		$1 + 2^{3} + 2^{7} - 2^{10} - 2^{37} + 2^{50}$	6	491 / 190	385 / 192
		$1 + 2^{3} + 2^{26} - 2^{44} + 2^{51}$	5	500 / 192	393 / 196
	$x' \equiv 47, 79 \bmod 112$ $T_1,\ a_2,\ D$	$-1 - 2^{2} - 2^{31} - 2^{35} + 2^{48}$	5	471 / 187	369 / 184
		$-1 - 2^{2} + 2^{20} - 2^{41} + 2^{47} - 2^{49}$	6	481 / 189	377 / 188
		$-1 - 2^{2} + 2^{11} - 2^{21} - 2^{35} + 2^{49}$	6	481 / 189	377 / 188
		$-1 - 2^{2} - 2^{4} - 2^{16} - 2^{26} - 2^{50}$	6*	491 / 190	385 / 192
KSS $k = 18$ (see §6.5)	$x' \equiv 4 \bmod 36$ $T_1,\ b_2,\ D$	$2^{18} + 2^{34} - 2^{45} - 2^{64}$	4	508 / 203	376 / 187
		$2^{12} + 2^{46} - 2^{51} - 2^{64}$	4	508 / 203	376 / 187
		$2^{28} + 2^{47} - 2^{51} + 2^{64}$	4	508 / 203	376 / 187
		$2^{5} - 2^{15} + 2^{42} - 2^{65}$	4	516 / 205	382 / 190
	$x' \equiv 16 \bmod 108$ $T_1,\ b_6,\ M$	$2^{20} - 2^{24} + 2^{28} + 2^{35} - 2^{64}$	5	508 / 203	376 / 187
		$2^{4} - 2^{8} - 2^{23} + 2^{39} - 2^{64}$	5	508 / 203	376 / 187
		$-2^{13} - 2^{31} - 2^{44} - 2^{62} - 2^{64}$	5*	511 / 204	378 / 188
		$2^{22} - 2^{36} - 2^{38} - 2^{63} + 2^{65}$	5	513 / 204	380 / 189
		$-2^{15} - 2^{20} + 2^{45} + 2^{63} - 2^{65}$	5	513 / 204	380 / 189
		$-2^{12} + 2^{25} - 2^{60} + 2^{62} - 2^{65}$	5	515 / 205	381 / 190
		$2^{18} + 2^{29} - 2^{35} + 2^{37} + 2^{65}$	5	516 / 205	382 / 190
		$2^{7} - 2^{16} + 2^{40} + 2^{60} + 2^{65}$	5	516 / 205	382 / 190
		$-2^{5} + 2^{21} + 2^{35} - 2^{62} - 2^{65}$	5	517 / 205	383 / 191
		$-2^{24} - 2^{31} + 2^{43} + 2^{62} + 2^{65}$	5	517 / 205	383 / 191
		$2^{25} + 2^{34} + 2^{37} + 2^{64} - 2^{66}$	5	521 / 206	386 / 192
		$-2^{3} - 2^{32} - 2^{42} - 2^{64} + 2^{66}$	5	521 / 206	386 / 192
	$x' \equiv 79 \bmod 108$ $T_1,\ M,\ b_3$	$2^{1} + 2^{29} + 2^{59} + 2^{65}$	4*	516 / 205	382 / 190
	$x' \equiv 7, 43 \bmod 108$ $T_1,\ M,\ b_{-4}$	$2^{1} - 2^{15} + 2^{18} + 2^{63} + 2^{65}$	5	519 / 205	384 / 191

Table B.2: Low weight curves offering 192-bit security.

224-bit secure curves					
family	subfamily/details/rating	x_0	weight	$\mathbb{F}_q$ (bits) / $\mathbb{F}_{q^k}$ sec.	r (bits) / $E[r]$ sec.
BLS $k = 12$ (see §6.3)	$x \equiv 64 \bmod 72$ $T_1,\ b_{-2},\ D$	$-2^{34} + 2^{58} + 2^{150}$	3	899 / 219	601 / 300
		$2^{76} + 2^{95} - 2^{151}$	3	905 / 219	604 / 301
		$-2^{101} - 2^{138} - 2^{151}$	3*	905 / 219	605 / 302
		$2^{43} - 2^{59} + 2^{152}$	3	911 / 220	608 / 303
		$-2^{5} - 2^{61} + 2^{153}$	3	917 / 221	612 / 305
		$2^{50} - 2^{131} - 2^{154}$	3	923 / 221	617 / 308
		$2^{96} - 2^{131} + 2^{155}$	3	929 / 222	620 / 309
		$2^{7} - 2^{95} - 2^{155}$	3	929 / 222	621 / 310
		$-2^{65} + 2^{77} + 2^{156}$	3	935 / 222	625 / 312
		$-2^{75} + 2^{111} + 2^{156}$	3	935 / 222	625 / 312
		$2^{30} - 2^{59} - 2^{158}$	3	947 / 224	633 / 316
		$-2^{21} + 2^{60} + 2^{159}$	3	953 / 224	637 / 318
	$x \equiv 16, 88 \bmod 216$ $T_1,\ b_4,\ M$	$-2^{51} + 2^{88} - 2^{150}$	3	899 / 219	600 / 299
		$2^{88} + 2^{91} - 2^{151}$	3	905 / 219	604 / 301
		$2^{22} + 2^{46} + 2^{151}$	3*	905 / 219	605 / 302
		$-2^{105} - 2^{124} + 2^{152}$	3	911 / 220	608 / 303
		$-2^{47} + 2^{144} - 2^{154}$	3	923 / 221	616 / 307
		$-2^{4} + 2^{88} + 2^{154}$	3	923 / 221	617 / 308
		$-2^{7} + 2^{137} - 2^{155}$	3	929 / 222	620 / 309
		$-2^{127} + 2^{140} + 2^{155}$	3	929 / 222	621 / 310
		$-2^{27} - 2^{147} + 2^{155}$	3	929 / 222	620 / 309
	$x \equiv 160 \bmod 216$ $T_1,\ b_{-3},\ M$	$-2^{95} + 2^{116} - 2^{150}$	3	899 / 219	600 / 299
		$-2^{59} - 2^{67} - 2^{152}$	3*	911 / 220	609 / 304
		$2^{22} - 2^{69} + 2^{153}$	3	917 / 221	612 / 305
		$-2^{22} - 2^{35} - 2^{153}$	3*	917 / 221	613 / 306
		$-2^{83} - 2^{150} - 2^{155}$	3*	929 / 222	621 / 310
		$2^{14} - 2^{34} - 2^{159}$	3	953 / 224	637 / 318
		$-2^{89} - 2^{100} - 2^{159}$	3*	953 / 224	637 / 318
KSS $k = 16$ (see §6.4)	$x' \equiv 61, 93 \bmod 112$ $T_1,\ a_1,\ M$	$1 - 2^{22} + 2^{56} + 2^{66} - 2^{69}$	5	679 / 219	535 / 267
		$1 + 2^{14} + 2^{17} + 2^{36} + 2^{69}$	5	681 / 220	537 / 268
		$1 + 2^{25} - 2^{33} - 2^{65} - 2^{70}$	5	691 / 221	545 / 272
		$1 - 2^{20} - 2^{62} + 2^{69} - 2^{71}$	5	696 / 222	549 / 274
		$1 - 2^{47} - 2^{54} - 2^{65} + 2^{71}$	5	700 / 222	552 / 275
		$1 + 2^{21} - 2^{38} + 2^{51} + 2^{71}$	5	701 / 222	553 / 276
		$1 + 2^{23} + 2^{48} + 2^{57} + 2^{71}$	5*	701 / 222	553 / 276
		$1 - 2^{15} - 2^{55} - 2^{66} - 2^{72}$	5	711 / 224	561 / 280
	$x' \equiv 5, 37 \bmod 112$ $T_1,\ a_1,\ D$	$1 - 2^{3} + 2^{12} + 2^{22} + 2^{30} - 2^{69}$	6	681 / 220	537 / 268
		$1 - 2^{3} + 2^{11} - 2^{47} - 2^{71}$	5	701 / 222	553 / 276
KSS $k = 18$ (see §6.5)	$x' \equiv 4 \bmod 36$ $T_1,\ b_2,\ D$	$2^{20} + 2^{26} + 2^{36} - 2^{76}$	4	604 / 219	448 / 223
		$2^{31} - 2^{36} + 2^{51} - 2^{76}$	4	604 / 219	448 / 223
		$2^{38} + 2^{41} + 2^{62} + 2^{76}$	4*	604 / 219	448 / 223
		$2^{6} + 2^{18} - 2^{39} - 2^{78}$	4	620 / 222	460 / 229
		$-2^{19} - 2^{45} + 2^{50} - 2^{78}$	4	620 / 222	460 / 229
		$2^{18} - 2^{57} + 2^{61} - 2^{79}$	4	628 / 223	466 / 232
		$2^{7} - 2^{24} - 2^{26} - 2^{80}$	4	636 / 224	472 / 235
		$-2^{3} - 2^{18} - 2^{49} + 2^{80}$	4	636 / 224	472 / 235
		$2^{24} - 2^{40} + 2^{56} + 2^{80}$	4	636 / 224	472 / 235
		$2^{6} - 2^{13} - 2^{73} - 2^{80}$	4	636 / 224	472 / 235
	$x' \equiv 16 \bmod 108$ $T_1,\ b_6,\ M$	$2^{18} - 2^{40} - 2^{66} + 2^{74} - 2^{76}$	5	601 / 219	446 / 222
		$-2^{30} + 2^{40} + 2^{50} + 2^{76}$	4	604 / 219	448 / 223
		$2^{13} + 2^{41} + 2^{58} + 2^{71} + 2^{76}$	5*	604 / 219	448 / 223
		$-2^{15} + 2^{18} - 2^{26} - 2^{72} - 2^{76}$	5	605 / 220	449 / 224
		$2^{13} + 2^{23} - 2^{28} + 2^{72} + 2^{76}$	5	605 / 220	449 / 224
		$-2^{6} - 2^{24} - 2^{37} + 2^{75} - 2^{77}$	5	609 / 220	452 / 225
		$-2^{20} - 2^{62} + 2^{68} + 2^{75} - 2^{77}$	5	609 / 220	452 / 225
		$2^{20} + 2^{32} - 2^{62} - 2^{77}$	4	612 / 221	454 / 226
		$-2^{13} + 2^{31} + 2^{37} + 2^{54} + 2^{77}$	5	612 / 221	454 / 226
	$x' \equiv 7, 43 \bmod 108$ $T_1,\ b_{-4},\ M$	$2^{1} + 2^{17} - 2^{22} + 2^{30} + 2^{76}$	5	604 / 219	448 / 223
		$2^{1} - 2^{14} - 2^{55} - 2^{63} - 2^{76}$	5	604 / 219	448 / 223
		$2^{1} + 2^{14} - 2^{61} - 2^{64} - 2^{76}$	5	604 / 219	448 / 223
	$x' \equiv 79 \bmod 108$ $T_1,\ b_3,\ M$	$2^{1} + 2^{7} - 2^{11} - 2^{41} + 2^{77}$	5	612 / 221	454 / 226

Table B.3: Low weight curves offering 224-bit security.

256-bit secure curves					
	subfamily/details	x_0	weight	$\mathbb{F}_q$ (bits) / $\mathbb{F}_{q^k}$ sec.	r (bits) / $E[r]$ sec.
BLS $k = 27$ (see §6.6)	$x \equiv 5, 14, 32 \bmod 36$ $T_1,\ b_{-3},\ M$	$-2^{11} - 2^{15} - 2^{23} - 2^{26}$	4^*	522 / 245	470 / 234
		$-2^4 - 2^7 + 2^{21} - 2^{25} + 2^{27}$	5	531 / 247	478 / 238
		$2^2 + 2^7 - 2^{18} - 2^{21} + 2^{27}$	5	538 / 249	484 / 241
		$2^6 - 2^{12} - 2^{17} - 2^{27}$	4	539 / 249	485 / 242
		$2^4 + 2^8 + 2^{16} - 2^{23} - 2^{27}$	5	541 / 249	486 / 242
		$-2^2 + 2^{11} + 2^{14} - 2^{24} - 2^{27}$	5	542 / 249	488 / 243
		$2^9 + 2^{19} - 2^{21} - 2^{26} + 2^{28}$	5	550 / 251	495 / 247
		$-2^4 - 2^{12} + 2^{24} - 2^{26} + 2^{28}$	5	553 / 252	498 / 248
		$2^4 - 2^7 + 2^{14} - 2^{25} + 2^{28}$	5	555 / 252	499 / 249
		$2^3 + 2^7 - 2^{19} - 2^{24} + 2^{28}$	5	557 / 252	501 / 250
		$-2^2 - 2^9 + 2^{11} + 2^{17} + 2^{28}$	5	559 / 253	503 / 251
		$-2^6 + 2^{11} + 2^{13} + 2^{24} + 2^{28}$	5	561 / 253	504 / 251
		$-2^1 - 2^4 + 2^{22} - 2^{26} - 2^{28}$	5	565 / 254	508 / 253
		$-2^{11} + 2^{14} + 2^{20} + 2^{26} + 2^{28}$	5	565 / 254	509 / 254
		$-2^3 - 2^5 + 2^{12} - 2^{14} + 2^{27} - 2^{29}$	6	571 / 255	513 / 256
		$2^6 - 2^{13} + 2^{19} + 2^{22} - 2^{27} + 2^{29}$	6	571 / 255	514 / 256
		$2^{10} + 2^{12} - 2^{18} - 2^{23} + 2^{27} - 2^{29}$	6	571 / 255	514 / 256
		$2^1 + 2^5 + 2^{15} + 2^{26} - 2^{29}$	5	575 / 256	517 / 258
	$x \equiv 11, ..., 1235 \bmod 1260$ $T_1,\ b_9,\ D$	$-1 + 2^7 + 2^{14} + 2^{23} - 2^{27}$	5	537 / 248	483 / 241
		$-1 - 2^9 - 2^{14} + 2^{16} + 2^{27}$	5	539 / 249	485 / 242
		$2^1 + 2^5 + 2^{15} - 2^{25} + 2^{28}$	5	555 / 252	499 / 249
		$2^4 + 2^7 + 2^{22} + 2^{25} - 2^{28}$	5	555 / 252	499 / 249
		$-2^1 + 2^5 - 2^{21} + 2^{23} - 2^{28}$	5	558 / 253	502 / 250
		$-2^2 + 2^5 + 2^{15} - 2^{28}$	4	559 / 253	503 / 251
		$2^3 - 2^{11} + 2^{17} + 2^{23} + 2^{28}$	5	560 / 253	504 / 251
		$1 - 2^{10} + 2^{13} + 2^{20} + 2^{26} + 2^{28}$	6	565 / 254	509 / 254
		$1 + 2^9 - 2^{13} - 2^{15} + 2^{27} - 2^{29}$	6	571 / 255	513 / 256
		$1 + 2^2 - 2^{10} + 2^{18} + 2^{27} - 2^{29}$	6	571 / 255	513 / 256
	$x \equiv 23 \bmod 36$ $T_1,\ b_3,\ M$	$-1 + 2^8 + 2^{13} - 2^{15} + 2^{27}$	5	539 / 249	485 / 242
		$-1 - 2^8 - 2^{20} - 2^{24} + 2^{26} - 2^{28}$	6	553 / 252	498 / 248
		$-1 - 2^7 - 2^{17} - 2^{21} + 2^{24} - 2^{28}$	6	557 / 252	501 / 250
		$-1 + 2^2 + 2^5 + 2^{10} + 2^{12} + 2^{28}$	6	559 / 253	503 / 251
		$-1 + 2^2 - 2^{21} - 2^{24} + 2^{26} + 2^{28}$	6	564 / 254	507 / 253
		$-1 - 2^6 + 2^{14} - 2^{20} - 2^{26} - 2^{28}$	6	565 / 254	509 / 254
		$-1 - 2^{11} - 2^{17} - 2^{21} - 2^{27} + 2^{29}$	6	570 / 255	513 / 256
	$x \equiv 110, ..., 1244(1260)$ $T_1,\ b_7,\ D$	$-1 - 2^2 - 2^6 + 2^{27}$	4	539 / 249	485 / 242
		$-2^3 - 2^7 + 2^{19} - 2^{27}$	4	539 / 249	485 / 242
		$1 - 2^{11} - 2^{18} + 2^{20} - 2^{26} + 2^{28}$	6	551 / 251	496 / 247
		$1 - 2^{13} + 2^{15} - 2^{21} + 2^{24} - 2^{28}$	6	557 / 252	501 / 250
		$1 - 2^9 - 2^{12} + 2^{14} + 2^{20} - 2^{28}$	6	559 / 253	503 / 251
	$x \equiv 2, ..., 1136 \bmod 1260$ $T_1,\ b_{-7},\ D$	$2^1 + 2^8 - 2^{14} + 2^{22} - 2^{28}$	5	558 / 253	503 / 251
		$1 - 2^4 - 2^6 + 2^{25} + 2^{28}$	5	562 / 253	506 / 252

Table B.4: Low weight curves offering 256-bit security.

288-bit secure curves					
	subfamily/details	x_0	weight	$\mathbb{F}_q$ (bits) / $\mathbb{F}_{q^k}$ sec.	r (bits) / $E[r]$ sec.
BLS $k=27$ (see §6.6)	$x \equiv 5, 14, 32 \bmod 36$ $T_1,\ b_{-3},\ M$	$1-2^4-2^9+2^{36}$	4	719 / 281	647 / 323
		$1+2^{20}+2^{26}-2^{30}-2^{35}+2^{37}$	6	730 / 283	657 / 328
		$1+2^8+2^{12}+2^{19}-2^{35}+2^{37}$	6	731 / 283	657 / 328
		$1-2^2+2^{16}+2^{32}-2^{35}+2^{37}$	6	732 / 284	659 / 329
		$1+2^2-2^{11}+2^{33}-2^{35}+2^{37}$	6	733 / 284	660 / 329
		$1-2^{10}-2^{18}-2^{24}-2^{34}+2^{37}$	6	735 / 284	661 / 330
		$1+2^{17}+2^{20}+2^{24}-2^{33}+2^{37}$	6	737 / 284	663 / 331
		$1+2^{11}+2^{15}+2^{17}+2^{31}+2^{37}$	6^*	739 / 285	665 / 332
		$2^{10}+2^{12}-2^{17}+2^{37}$	4	739 / 285	665 / 332
		$1+2^7-2^9+2^{15}+2^{37}$	5	739 / 285	665 / 332
		$2^7+2^{24}-2^{29}-2^{37}$	4	739 / 285	665 / 332
		$1+2^2+2^{18}-2^{22}+2^{32}+2^{37}$	6	740 / 285	666 / 332
		$1+2^{10}+2^{17}+2^{21}-2^{35}-2^{37}$	6	745 / 286	671 / 335
		$-2^9+2^{24}-2^{36}+2^{38}$	4	751 / 287	675 / 337
		$1-2^{12}-2^{20}-2^{32}+2^{38}$	5	758 / 288	683 / 341
		$1+2^8-2^{25}-2^{31}+2^{38}$	5	759 / 288	683 / 341
		$2^{26}-2^{35}-2^{37}+2^{39}$	4	768 / 289	691 / 345
	$x \equiv 11, \ldots, 1235 \bmod 1260$ $T_1,\ b_9,\ D$	$-1+2^5+2^7-2^{26}-2^{31}+2^{37}$	6	738 / 285	664 / 331
		$1+2^8-2^{12}+2^{17}+2^{37}$	5	739 / 285	665 / 332
		$-2^1-2^{14}-2^{20}+2^{24}+2^{37}$	5	739 / 285	665 / 332
		$2^5-2^{17}-2^{21}-2^{24}+2^{37}$	5	739 / 285	665 / 332
		$2^5-2^9+2^{25}-2^{27}+2^{37}$	5	739 / 285	665 / 332
		$1+2^3+2^{15}+2^{19}+2^{27}+2^{37}$	6^*	739 / 285	665 / 332
		$2^6+2^{11}+2^{27}-2^{29}+2^{37}$	5	739 / 285	665 / 332
		$2^1+2^5-2^{15}+2^{29}-2^{37}$	5	739 / 285	665 / 332
		$-1+2^{16}-2^{26}-2^{34}-2^{37}$	5	742 / 285	668 / 333
		$2^8+2^{24}-2^{33}-2^{38}$	4	760 / 288	684 / 341
	$x \equiv 23 \bmod 36$ $T_1,\ b_3,\ M$	$-1+2^{13}+2^{23}+2^{28}+2^{36}$	5	719 / 281	647 / 323
		$-1-2^{25}-2^{29}-2^{31}-2^{35}+2^{37}$	6	730 / 283	657 / 328
		$-1+2^7-2^{16}+2^{22}-2^{34}+2^{37}$	6	735 / 284	661 / 330
		$-1-2^{10}-2^{22}-2^{29}-2^{37}$	5^*	739 / 285	665 / 332
		$-1+2^7-2^{13}-2^{30}-2^{37}$	5	739 / 285	665 / 332
		$-1+2^{15}+2^{29}-2^{31}+2^{38}$	5	759 / 288	683 / 341
		$-1-2^2-2^8-2^{21}-2^{30}-2^{38}$	6^*	759 / 288	683 / 341
		$-1+2^{15}+2^{29}-2^{31}+2^{38}$	5	759 / 288	683 / 341
		$-1-2^8-2^{11}-2^{22}+2^{38}$	5	759 / 288	683 / 341
	$x \equiv 110, \ldots, 1244 \bmod 1260$ $T_1,\ b_7,\ D$	$-2^6-2^{20}-2^{27}-2^{30}-2^{37}$	5^*	739 / 285	665 / 332
		$-2^2-2^{20}-2^{26}-2^{31}-2^{37}$	5^*	739 / 285	665 / 332
		$-1+2^2-2^6-2^{26}+2^{38}$	5^*	759 / 288	683 / 341
	$x \equiv 38, \ldots, 1253 \bmod 1260$ $T_1,\ b_{-5},\ D$	$2^7+2^{13}-2^{37}$	3	739 / 285	665 / 332
		$1-2^4-2^8-2^{27}+2^{37}$	5	739 / 285	665 / 332
		$2^7+2^9+2^{22}-2^{26}-2^{37}$	5	739 / 285	665 / 332
		$-2^1+2^8+2^{11}-2^{28}+2^{37}$	5	739 / 285	665 / 332
		$2^8+2^{15}-2^{18}+2^{28}+2^{37}$	5	739 / 285	665 / 332
		$1+2^{16}+2^{33}+2^{38}$	4^*	760 / 288	684 / 341
KSS $k=32$ (see §6.7)	$x' \equiv 453, 981 \bmod 3824$ $T_1,\ a_1,\ D$	$1+2^{14}+2^{17}+2^{21}+2^{30}-2^{32}+2^{37}-2^{39}$	8	674 / 294	572 / 285
		$1-2^5+2^{10}+2^{12}-2^{18}-2^{37}+2^{39}$	7	674 / 294	571 / 285
		$1+2^6-2^{14}-2^{21}-2^{30}+2^{32}+2^{35}-2^{39}$	8	679 / 295	576 / 287
		$1-2^7-2^9+2^{21}+2^{26}-2^{28}-2^{30}+2^{39}$	8	681 / 296	578 / 288
		$1+2^9-2^{11}+2^{14}+2^{19}-2^{21}-2^{24}-2^{39}$	8	681 / 296	578 / 288
		$1+2^7-2^9-2^{14}-2^{17}-2^{20}-2^{29}-2^{39}$	8	681 / 296	578 / 288
		$1-2^7-2^9+2^{21}+2^{26}-2^{28}-2^{30}+2^{39}$	8	681 / 296	578 / 288
		$1-2^7-2^{12}+2^{17}+2^{22}-2^{28}-2^{35}-2^{39}$	8	683 / 296	580 / 289
		$1-2^{23}-2^{32}+2^{35}+2^{39}$	5	682 / 296	579 / 289
	$x' \equiv 2365, 2893 \bmod 3824$ $T_1,\ a_1,\ M$	$1-2^3-2^7+2^{25}-2^{36}+2^{38}$	6	656 / 291	555 / 277
		$1-2^3+2^7-2^{12}-2^{28}-2^{31}-2^{35}-2^{39}$	8	683 / 296	580 / 289

Table B.5: Low weight curves offering 288-bit security.

320-bit secure curves					
	subfamily/details	x_0	weight	$\mathbb{F}_q$ (bits) / $\mathbb{F}_{q^k}$ sec.	r (bits) / $E[r]$ sec.
KSS $k = 32$ (see §6.7)	$x' \equiv 453, 981 \bmod 3824$ $T_1,\ a_1,\ D$	$1 - 2^9 - 2^{13} - 2^{28} + 2^{31} + 2^{40} - 2^{45}$	7	788 / 314	673 / 336
		$1 - 2^7 - 2^{10} - 2^{19} + 2^{35} + 2^{40} - 2^{45}$	7	788 / 314	673 / 336
		$1 + 2^9 + 2^{12} - 2^{15} + 2^{21} - 2^{23} - 2^{25} + 2^{45}$	8	789 / 315	674 / 336
		$1 + 2^4 + 2^6 - 2^{18} + 2^{26} - 2^{28} - 2^{34} + 2^{45}$	8	789 / 315	674 / 336
		$1 - 2^6 + 2^{14} - 2^{20} - 2^{22} + 2^{34} - 2^{46}$	7	807 / 318	690 / 344
		$1 + 2^5 + 2^{17} - 2^{25} + 2^{29} - 2^{36} + 2^{46}$	7	807 / 318	690 / 344
		$1 + 2^6 + 2^8 - 2^{13} - 2^{18} - 2^{36} - 2^{47}$	7	825 / 320	706 / 352
		$1 - 2^8 - 2^{18} + 2^{24} + 2^{37} - 2^{46} + 2^{48}$	7	836 / 322	715 / 357
KSS $k = 36$ (see §6.8)	$x' \equiv 1376, 1880 \bmod 2664$ $T_1,\ b_2,\ D$	$-2^3 - 2^{17} - 2^{28} - 2^{52} + 2^{55}$	5	753 / 324	631 / 315
		$2^3 + 2^{14} - 2^{23} + 2^{34} - 2^{36} - 2^{55}$	6	756 / 325	633 / 316
		$2^5 + 2^9 + 2^{26} - 2^{31} + 2^{40} - 2^{55}$	6	756 / 325	633 / 316
		$-2^9 + 2^{25} - 2^{27} + 2^{38} + 2^{42} + 2^{55}$	6	756 / 325	633 / 316
		$-2^3 + 2^{12} + 2^{14} - 2^{33} + 2^{43} + 2^{55}$	6	756 / 325	633 / 316
		$-2^3 - 2^8 - 2^{29} - 2^{34} - 2^{45} - 2^{55}$	6*	756 / 325	633 / 316
		$-2^8 - 2^{21} - 2^{31} + 2^{42} + 2^{45} - 2^{55}$	6	756 / 325	633 / 316
		$-2^7 - 2^{19} - 2^{32} + 2^{44} - 2^{48} - 2^{55}$	6	756 / 325	633 / 316
	$x' \equiv 104, \ldots, 7592 \bmod 7992$ $T_1,\ b_{-4},\ M$	$-2^3 + 2^{12} - 2^{15} + 2^{24} - 2^{34} + 2^{55}$	6	756 / 325	633 / 316
		$-2^5 + 2^8 - 2^{11} + 2^{35} + 2^{37} - 2^{55}$	6	756 / 325	633 / 316
		$-2^7 - 2^{25} + 2^{31} - 2^{34} - 2^{39} - 2^{55}$	6	756 / 325	633 / 316
		$-2^{11} + 2^{21} + 2^{36} + 2^{42} - 2^{44} - 2^{55}$	6	756 / 325	633 / 316
		$2^3 + 2^7 - 2^{24} - 2^{33} + 2^{47} + 2^{55}$	6	756 / 325	633 / 316
	$x' \equiv 2768, 4928 \bmod 7992$ $T_1,\ b_3,\ M$	$2^{10} - 2^{17} + 2^{19} - 2^{21} - 2^{51} + 2^{55}$	6	754 / 324	632 / 315
		$2^4 + 2^7 + 2^{32} - 2^{48} + 2^{52} - 2^{55}$	6	753 / 324	631 / 315
		$-2^{30} - 2^{33} - 2^{40} - 2^{48} - 2^{50} + 2^{55}$	6	755 / 324	633 / 316
		$2^{12} + 2^{32} + 2^{41} - 2^{45} + 2^{50} + 2^{55}$	6	756 / 325	634 / 316
		$-2^6 - 2^{19} - 2^{44} + 2^{47} + 2^{51} + 2^{55}$	6	757 / 325	634 / 316
		$2^3 + 2^{15} + 2^{17} + 2^{24} - 2^{29} + 2^{40} + 2^{55}$	7	756 / 325	633 / 316
		$-2^5 + 2^{11} - 2^{24} - 2^{30} + 2^{39} - 2^{41} - 2^{55}$	7	756 / 325	633 / 316
		$-2^5 - 2^{16} + 2^{18} - 2^{21} - 2^{31} - 2^{41} - 2^{55}$	7	756 / 325	633 / 316
		$-2^{15} + 2^{22} + 2^{27} - 2^{31} - 2^{33} - 2^{41} - 2^{55}$	7	756 / 325	633 / 316
		$2^{11} + 2^{16} + 2^{23} + 2^{34} - 2^{37} + 2^{56}$	6	770 / 327	645 / 322
		$2^6 + 2^{10} + 2^{31} + 2^{34} + 2^{50} - 2^{56}$	6	769 / 327	645 / 322
		$-2^{27} - 2^{34} + 2^{39} - 2^{44} - 2^{50} - 2^{56}$	6	770 / 327	646 / 322
		$2^3 - 2^6 - 2^{34} + 2^{39} + 2^{53} + 2^{56}$	6	772 / 327	647 / 323

Table B.6: Low weight curves offering 320-bit security.

352-bit secure curves					
	subfamily/details	x_0	weight	$\mathbb{F}_q$ (bits) / $\mathbb{F}_{q^k}$ sec.	r (bits) / $E[r]$ sec.
KSS $k=36$ (see §6.8)	$x' \equiv 2768, 4928 \bmod 7992$ $T_1,\ b_3,\ M$	$-2^{12}-2^{32}+2^{34}-2^{45}+2^{65}$	5	896 / 348	753 / 376
	$x' \equiv 1376, 1880 \bmod 2664$ $T_1,\ b_2,\ D$	$-2^5-2^{16}-2^{18}+2^{27}+2^{31}+2^{65}$	6	896 / 348	753 / 376
		$-2^{10}+2^{23}+2^{42}+2^{57}-2^{65}$	5	896 / 348	753 / 376
		$-2^{26}+2^{41}-2^{45}+2^{62}+2^{65}$	5	898 / 349	755 / 377
		$2^{21}-2^{31}+2^{49}-2^{58}+2^{66}$	5	910 / 351	765 / 382
	$x' \equiv 104, ..., 7592 \bmod 7992$ $T_1,\ b_{-4},\ M$	$-2^5-2^{16}-2^{46}+2^{50}+2^{65}$	5	896 / 348	753 / 376
		$2^{10}+2^{16}-2^{23}-2^{30}+2^{32}+2^{65}$	6	896 / 348	753 / 376
	$x' \equiv 821, 1325 \bmod 2664$ $T_1,\ b_{-1},\ M$	$-1+2^2+2^{33}+2^{37}+2^{43}-2^{55}+2^{66}$	7	910 / 351	765 / 382
		$-1+2^2+2^7-2^9+2^{19}+2^{28}-2^{66}$	7	910 / 351	765 / 382
		$-1+2^2-2^{19}-2^{31}-2^{49}+2^{52}+2^{66}$	7	910 / 351	765 / 382
	$x \equiv 437, 2597 \bmod 2664$ $T_1,\ b_{-1},\ D$	$-1+2^2+2^5-2^{21}+2^{45}+2^{54}-2^{66}$	7	910 / 351	765 / 382
		$-1-2^2+2^6+2^{18}-2^{22}-2^{33}-2^{66}$	7	910 / 351	765 / 382
BLS $k=48$ (see §6.9)	$x \equiv 16, 88 \bmod 216$ $T_1,\ b_4,\ M$	$-2^{11}-2^{21}+2^{43}$	3	773 / 369	688 / 343
		$-2^{30}-2^{36}-2^{38}+2^{44}$	4	790 / 373	704 / 351
		$2^3-2^{11}-2^{21}-2^{24}+2^{44}$	5	791 / 373	704 / 351
		$2^9+2^{26}-2^{42}-2^{44}$	4	797 / 374	710 / 354
	$x \equiv 64 \bmod 72$ $T_1,\ b_{-2},\ D$	$-2^3+2^8-2^{13}+2^{19}-2^{44}$	5	791 / 373	704 / 351
		$-2^7+2^{10}+2^{17}+2^{20}-2^{44}$	5	791 / 373	704 / 351
		$-2^8-2^{15}-2^{17}+2^{23}+2^{44}$	5	791 / 373	705 / 352
		$2^{12}-2^{14}+2^{17}+2^{26}+2^{44}$	5	791 / 373	705 / 352
	$x \equiv 160 \bmod 216$ $T_1,\ b_{-3},\ M$	$-2^{15}+2^{29}+2^{31}-2^{41}+2^{44}$	5	787 / 372	701 / 350
		$-2^{20}-2^{26}-2^{38}+2^{41}-2^{44}$	5	788 / 372	702 / 350
		$2^{11}+2^{18}+2^{27}-2^{31}+2^{44}$	5	791 / 373	704 / 351
		$-2^4-2^{18}-2^{21}+2^{24}+2^{44}$	5	791 / 373	705 / 352
		$-2^8+2^{13}-2^{25}+2^{34}+2^{44}$	5	791 / 373	705 / 352
		$-2^8+2^{15}+2^{30}+2^{40}+2^{44}$	5	792 / 373	706 / 352
		$-2^7+2^{25}-2^{29}+2^{43}-2^{45}$	5	801 / 375	714 / 356
	$x \equiv 7 \bmod 72$ $T_1,\ b_1,\ D$	$-1+2^3+2^{33}+2^{41}-2^{44}$	5	787 / 372	701 / 350
		$-1+2^6-2^{23}+2^{28}-2^{44}$	5	791 / 373	704 / 351
		$-1-2^6-2^{11}-2^{25}-2^{28}-2^{44}$	6*	791 / 373	705 / 352
		$-1+2^{18}-2^{25}+2^{29}+2^{44}$	5	791 / 373	705 / 352
		$-1-2^{30}-2^{37}+2^{40}+2^{44}$	5	792 / 373	706 / 352
		$-1+2^{18}-2^{25}+2^{29}+2^{44}$	5	791 / 373	705 / 352
		$-1-2^{14}-2^{17}-2^{26}-2^{31}-2^{44}$	6*	791 / 373	705 / 352
	$x \equiv 31 \bmod 72$ $T_1,\ b_1, M$	$-1+2^4-2^{15}-2^{19}-2^{23}+2^{44}$	6	791 / 373	704 / 351
		$-1-2^7-2^{10}-2^{13}-2^{16}-2^{44}$	6*	791 / 373	705 / 352
		$-1-2^{13}+2^{18}-2^{27}-2^{44}$	5	791 / 373	705 / 352
		$-1+2^{17}-2^{19}+2^{40}+2^{44}$	5	792 / 373	706 / 352

Table B.7: Low weight curves offering 352-bit security.

384-bit secure curves					
	subfamily/details	x_0	weight	$\mathbb{F}_q$ (bits) / $\mathbb{F}_{q^k}$ sec.	r (bits) / $E[r]$ sec.
BLS $k=48$ (see §6.9)	$x \equiv 64 \bmod 72$ $T_1,\ b_{-2},\ D$	$2^7+2^{17}+2^{32}-2^{48}$	4	863 / 387	768 / 383
		$-2^{17}+2^{30}-2^{35}+2^{48}$	4	863 / 387	768 / 383
		$-2^6-2^{22}+2^{36}-2^{48}$	4	863 / 387	768 / 383
		$-2^4+2^{11}-2^{16}+2^{19}-2^{48}$	5	863 / 387	768 / 383
		$2^7+2^{35}+2^{47}-2^{49}$	4	873 / 388	778 / 388
	$x \equiv 160 \bmod 216$ $T_1,\ b_{-3},\ M$	$2^7-2^{10}+2^{16}-2^{29}+2^{48}$	5	863 / 387	768 / 383
		$2^6+2^{34}+2^{40}+2^{48}$	4*	863 / 387	769 / 384
	$x \equiv 7 \bmod 72$ $T_1,\ b_1,\ D$	$-1+2^4-2^{16}-2^{31}+2^{48}$	5	863 / 387	768 / 383
		$-1+2^{16}+2^{18}+2^{30}-2^{48}$	5	863 / 387	768 / 383
		$-1+2^{12}+2^{17}-2^{20}+2^{22}-2^{48}$	6	863 / 387	768 / 383
		$-1+2^6-2^{13}+2^{20}-2^{23}+2^{48}$	6	863 / 387	768 / 383
		$-1-2^{13}+2^{15}+2^{18}+2^{48}$	5	863 / 387	769 / 384
		$-1+2^8-2^{15}+2^{17}-2^{24}-2^{48}$	6	863 / 387	769 / 384
		$-1-2^{10}-2^{13}+2^{23}-2^{29}-2^{48}$	6	863 / 387	769 / 384
	$x \equiv 16, 88 \bmod 216$ $T_1,\ b_4,\ M$	$2^3+2^{14}+2^{17}-2^{19}+2^{48}$	5	863 / 387	768 / 383
		$2^3-2^9-2^{19}+2^{30}-2^{48}$	5	863 / 387	768 / 383
		$-2^4-2^{16}+2^{20}-2^{25}+2^{48}$	5	863 / 387	768 / 383
		$2^7-2^{12}-2^{17}-2^{27}+2^{48}$	5	863 / 387	768 / 383
		$2^5+2^{18}-2^{27}+2^{36}-2^{48}$	5	863 / 387	768 / 383
		$-2^5-2^{11}+2^{25}+2^{29}+2^{48}$	5	863 / 387	769 / 384
		$2^5-2^8+2^{15}-2^{36}-2^{48}$	5	863 / 387	769 / 384
	$x \equiv 31 \bmod 72$ $T_1,\ b_1,\ M$	$-1+2^4-2^8-2^{27}+2^{48}$	5	863 / 387	768 / 383
		$-1+2^{13}+2^{23}+2^{28}-2^{30}+2^{48}$	6	863 / 387	768 / 383
		$-1-2^{10}-2^{21}+2^{25}+2^{30}-2^{48}$	6	863 / 387	768 / 383
		$-1+2^9-2^{22}+2^{25}+2^{30}+2^{48}$	6	863 / 387	769 / 384
		$-1+2^7+2^{17}+2^{19}+2^{26}+2^{48}$	6	863 / 387	769 / 384

Table B.8: Low weight curves offering 384-bit security.

Appendix C

Proofs of family trees

For each family, we first give proofs for the towers T_i in the corresponding tree, before giving proofs for the correct curve constants a_i or b_i. The proofs for the curve constants sometimes need results that follow from the towers they are associated with, so rather than prove things twice, we occasionally rely on the reader to match congruences with their corresponding reciprocity results from the tower proofs.

C.1 Towers

Our proofs of the towers mostly make use of the following theorem.

Theorem C.1 (Benger-Scott [BS10], Theorem 4)**.** *Let $m > 1$, $n > 0$ be integers, q and odd prime and $\alpha \in \mathbb{F}_{q^n}^{\times}$. The binomial $x^m - \alpha$ is irreducible in $\mathbb{F}_{q^n}[x]$ if the following two conditions are satisfied:*

1. *Each prime factor p of m divides $q-1$ and $N_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\alpha) \in \mathbb{F}_q$ is not a p^{th} residue in $\mathbb{F}_q$;*
2. *If $m \equiv 0 \bmod 4$, then $q^n \equiv 1 \bmod 4$.*

The *Norm* of $\alpha \in \mathbb{F}_{q^n}$ over $\mathbb{F}_q$ is defined as

$$N_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\alpha) = \prod_{i=0}^{n-1} \alpha^{q^i}$$

We will only be using Theorem C.1 to prove irreducibility in $\mathbb{F}_{q^2}[x]$ or $\mathbb{F}_{q^3}[x]$, i.e. we only need to compute $N_{\mathbb{F}_{q^2}/\mathbb{F}_q}$ and $N_{\mathbb{F}_{q^3}/\mathbb{F}_q}$, which we abbreviate to $N_{2,1}$ and $N_{3,1}$ respectively. The norm computation usually requires a trivial (possibly repeated) application of Fermat's little theorem, so we omit the details to save space.

Whether towering up to $\mathbb{F}_{q^2}$ or $\mathbb{F}_{q^3}$, or towering beyond them to $\mathbb{F}_{q^k}$ then, the proofs all amount to showing quadratic or cubic non-reciprocity in $\mathbb{F}_q$. We write the quadratic and cubic characters of a as usual, i.e. $(\frac{a}{q})_2$ and $(\frac{a}{q})_3$ respectively, and for quadratic reciprocity, we use the following two results.

Proposition C.2 ([IR90], §5, Prop. 5.1.3)**.** 2 *is a quadratic residue modulo* q *iff* $q \equiv 1, 7 \bmod 8$.

Theorem C.3 ([IR90], §5, Theorem 2)**.** *Let* p *be an odd prime.*

(a) If $p \equiv 1 \bmod 4$*, then* p *is a quadratic residue modulo* p *iff* $q \equiv r \bmod p$*, where* r *is a quadratic residue modulo* p*.*

(b) If $p \equiv 3 \bmod 4$*, then* p *is a quadratic residue modulo* q *iff* $q \equiv \pm b^2 \bmod 4p$*, where* b *is an odd integer prime to* p*.*

For cubic reciprocity, we apply Euler's conjectures [Lem00], which were originally based on Fermat's observation that for $q \equiv 1 \bmod 3$, q can be written as $q = a^2+3b^2$, where a and b are unique up to sign. For our purposes, a more convenient formulation of Euler's conjectures (which are also special cases of Lehmer's result [Leh58]) can be made in the following theorem, by instead writing $4q$ as $4q = L^2 + 27M^2$, where L and M are unique up to sign ([IR90, Prop. 8.3.2]).

Theorem C.4 (Euler's conjectures [Lem00], Prop. 7.1 - 7.4)**.** *For* $q \equiv 1 \bmod 3$*, let* L *and* M *be the unique integers (up to sign) such that* $4q = L^2+27M^2$*. Then,*

$$(i): \quad \left(\frac{2}{q}\right)_3 = 1 \leftrightarrow L \equiv M \equiv 0 \bmod 2; \quad (ii): \quad \left(\frac{3}{q}\right)_3 = 1 \leftrightarrow M \equiv 0 \bmod 3;$$
$$(iii): \quad \left(\frac{5}{q}\right)_3 = 1 \leftrightarrow LM \equiv 0 \bmod 5; \qquad (iv): \quad \left(\frac{7}{q}\right)_3 = 1 \leftrightarrow LM \equiv 0 \bmod 7;$$

The convenience of analyzing the equation $4q = L^2 + 27M^2$ comes from the CM norm equation for curves with discriminant D: $4q = t^2 - Df^2$. Curves of discriminant $D = -3$ are the only curves requiring cubic reciprocity (extensions) in this work, so we can always write $4q = t^2 + 3f^2$ where $t = t(x)$ and $f = f(x)$

are given in the family parameterizations. Depending on the different cases for (t, f) mod 6, three different manipulations of the CM norm equation (taken from our proofs in Chapter 5) can be employed to write $4q = L^2 + 27M^2$, given below.

$$\begin{aligned} (i) \qquad & 4q = t^2 + 27\,(f/3)^2 \\ (ii) \qquad & 4q = \left(\frac{3f+t}{2}\right)^2 + 27\left(\frac{t-f}{6}\right)^2 \\ (iii) \qquad & 4q = \left(\frac{t-3f}{2}\right)^2 + 27\left(\frac{t+f}{6}\right)^2 \end{aligned} \tag{C.1}$$

Throughout the towering proofs we refer to equation (C.1)-(i),(ii), or (iii) depending on how L and M (in $4q = L^2 + 27M^2$) are computed from $t(x)$ and $f(x)$, which we abbreviate to t and f for short.

C.2 Curve equations

For $k = 8$, $k = 16$ and $k = 32$ KSS curves, the correct curve has CM discriminant $D = -1$ and is of the form $E/\mathbb{F}_q : y^2 = x^3 + ax$. If g is a fourth-power-free integer, then a is precisely one of $\{1, g, g^2, g^3\}$ ([Sil09, §X.6]). For all the other families, the correct curve has discriminant $D = -3$ and is of the form $E/\mathbb{F}_q : y^2 = x^3 + b$. In this scenario, if g is neither square or cube in $\mathbb{F}_q$, then b is precisely one of $\{1, g, g^2, g^3, g^4, g^5\}$ ([Sil09, §X.5, Corr. 5.4.1]). For both of these special scenarios (CM discriminants), Rubin and Silverberg [RS10] present simple algorithms (Alg. C.1 and Alg. C.2 below) to determine the correct a or b value, both of which they say are "essentially due to Gauss". Our proofs make constant use of these algorithms.

Notice that the choice of a in Alg. C.1 is such that $y^2 = x^3 - ax$ is the correct curve, whilst we have been using $y^2 = x^3 + ax$ as the correct curve throughout. Thus, a specific proof that $a = \tilde{a}$ will actually use $-\tilde{a}$ when Alg. C.1 is invoked.

C.3 Proofs for each family

We shrink the proofs themselves for space considerations.

Algorithm C.1 The Rubin-Silverberg algorithm for finding the correct curve $y^2 = x^3 + ax$ [RS10, Alg 3.4].

Suppose $D = -1$, i.e. $4q = t^2 + f^2$ and $E/\mathbb{F}_q : y^2 = x^3 - ax$. Set $L = t/2$ and $M = f/2$. A correct curve (value of a) is found by the following algorithm.

- Step 1: If L is odd and $L - 1 \equiv M \bmod 4$, then $a = 1$.
- Step 2: If L is odd and $L - 1 \not\equiv M \bmod 4$, then $a \in \mathbb{F}_q$ is any square that is not a fourth power (i.e. $a^{(q-1)/4} \equiv -1 \bmod q$).
- Step 3: If L is even, replace M by $-M$ if necessary to ensure that $M-1 \equiv L \bmod 4$, then output any $a \in \mathbb{F}_q$ such that $a^{(q-1)/4} \equiv L/M \bmod q$.

Algorithm C.2 The Rubin-Silverberg algorithm for finding the correct curve $y^2 = x^3 + b$ [RS10, Alg 3.5].

Suppose $D = -3$, i.e. $4q = t^2 + 3f^2$ and $E/\mathbb{F}_q : y^2 = x^3 + b$. A correct curve (value of b) is found by the following algorithm.

- Step 1: If $f \equiv 0 \bmod 3$ and $t \equiv 2 \bmod 3$, then $b = 16$.
- Step 2: If $f \equiv 0 \bmod 3$ and $t \equiv 1 \bmod 3$, then $b = 16b'$, where $b' \in \mathbb{F}_q$ is any cube that is not a square (i.e. $b'^{(q-1)/6} \equiv -1 \bmod q$).
- Step 3: If $f \not\equiv 0 \bmod 3$. replace f by $-f$ if necessary to ensure that $f \equiv 1 \bmod 3$. If $t \equiv 2 \bmod 3$, output $b = 16b'$ for any b' satisfying $b'^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$.
- Step 4: Otherwise, output $b = 16b'$ for any b' satisfying $b'^{(q-1)/6} \equiv 2t/(3f + t) \bmod q$.

$k = 8$ Brezing-Weng curves

T_1: $x \equiv 1, 3, 9, 11 \bmod 16$ all imply $q \equiv 5 \bmod 8$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 2)$ by Prop. C.2. Now, $N_{2,1}(u) = 2$ and we already have $(\frac{2}{q})_2 = -1$, so that $x^4 - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_2: $x \equiv 5, 7, 13, 23 \bmod 24$ all imply $q \equiv 17 \bmod 24$. Using Theorem C.3-(b), with $q \equiv 5 \bmod 12$, and since the odd squares modulo 12 are either 1 or 9, we have that $(\frac{\pm 3}{q})_2 = -1$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 3)$. We also have that $N_{2,1}(u) = 3$, so that $x^4 - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_3: $x \equiv 21, ..., 117 \bmod 120$ all give $q \equiv 13, 17 \bmod 20$, invoking Theorem C.3-(b), and since the odd squares modulo 20 are either $1, 5$ or 9, we have that $(\frac{\pm 5}{q})_2 = -1$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 5)$. We also have that $N_{2,1}(u) = 5$, so that $x^4 - u$ is

irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

a_1: $x \equiv 1, 9 \bmod 16$ gives $(L, M) \equiv (1, 2) \bmod 4$, then Step 2 of Alg. C.1 equipped with $(\frac{-1}{q})_2 = 1$ but $(\frac{-1}{q})_4 = -1$ gives the result. □

a_2: $x \equiv 11 \bmod 16$ gives $(L, M) \equiv (2, 1) \bmod 4$, so replace M by $-M$ and use Alg. C.1-Step 3 to give $-2^{(q-1)/4} \equiv L/M \bmod q$. $x \equiv 13, 29 \bmod 48$ and $x \equiv 141, 189, 237 \bmod 240$ gives $(L, M) \equiv (3, 0) \bmod 4$, so Step 2 of Alg. C.1 this time equipped with $(\frac{-2}{q})_2 = 1$ but $(\frac{-2}{q})_4 = -1$ ([Sil09, Prop. 6.6]) gives the result. □

a_{-2}: $x \equiv 3 \bmod 16$ gives $(L, M) \equiv (2, 1) \bmod 4$, so replace M by $-M$ in Alg. C.1 - Step 3 and observe that $2^{(q-1)/4} \equiv L/M \bmod q$. □

a_3: $x \equiv 7 \bmod 24$ gives $(L, M) \equiv (0, 3) \bmod 4$, so replace M by $-M$ and use Alg. C.1-Step 3 and observe that $-3^{(q-1)/4} \equiv L/M \bmod q$. $x \equiv 21, 69, 117 \bmod 240$ gives $(L, M) \equiv (3, 0) \bmod 4$, so Step 2 of Alg. C.1 this time equipped with $(\frac{-3}{q})_2 = 1$ but $(\frac{-3}{q})_4 = -1$ gives the result. □

a_5: $x \equiv 11 \bmod 120$ and $x \equiv 71, 191 \bmod 240$ give $(L, M) \equiv (0, 3) \bmod 4$, so replace M by $-M$ and use Alg. C.1-Step 3 and observe that $-5^{(q-1)/4} \equiv L/M \bmod q$. $x \equiv 5, 85 \bmod 240$ gives $(L, M) \equiv (3, 0) \bmod 4$, so Step 2 of Alg. C.1 equipped with $(\frac{-5}{q})_2 = 1$ but $(\frac{-5}{q})_4 = -1$ gives the result. □

a_6: $x \equiv 47, 95, 143, 239 \bmod 240$ gives $(L, M) \equiv (0, 3) \bmod 4$, so replace M by $-M$ and use Alg. C.1-Step 3 and observe that $-6^{(q-1)/4} \equiv L/M \bmod q$.

$k = 12$ BLS curves

T_1: $x \equiv 7, 31, 64 \bmod 72$ and $160 \bmod 216$ all imply $q \equiv 19 \bmod 24$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 1)$. Note that $2, 3 \mid q - 1$, and $N_{2,1}(u + 1) = 2$, $(\frac{2}{q})_2 = -1$ (Prop. C.2) and $(\frac{2}{q})_3 = -1$ as follows. For $x \equiv 7 \bmod 72$, $t \equiv f \equiv 2 \bmod 6$, so use Equation (C.1)-(ii) and observe that both L and M are both odd. For $x \equiv 31 \bmod 72$, $t \equiv 2 \bmod 6$ and $f \equiv 4 \bmod 6$, so use Equation (C.1)-(iii) to see that L and M are both odd. For $x \equiv 64 \bmod 72$, Equation (C.1)-(i) yields this directly since f is a multiple of 3, say $f = 3M$, giving $4q = L^2 + 27M^2$, where $L = t = x + 1 \equiv 65 \bmod 72$ is odd. For $x \equiv 160 \bmod 216$, observe that $t \equiv f \equiv 5 \bmod 6$ so use Equation (C.1)-(ii) to further deduce that L and M are both odd. Thus $N_{2,1}(u + 1) = 2$, and $(\frac{2}{q})_3 = (\frac{2}{q})_2 = -1$ by Theorem C.4-(i), so that $v^6 - (u + 1)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_2: $x \equiv 55, 127, 343 \bmod 360$ imply $q \equiv 19 \bmod 144$, $x \equiv 43, ..., 307 \bmod 360$ imply $q \equiv 7 \bmod 24$, $x \equiv 28, 100, 172 \bmod 360$ implies $q \equiv 127 \bmod 216$, $x \equiv 124, ..., 1060 \bmod 1800$ implies $q \equiv 127 \bmod 360$, $x \equiv 4, ..., 1012 \bmod 1080$ implies $q \equiv 79 \bmod 108$. In all cases, $q \equiv 7 \bmod 12$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 1)$. $N_{2,1}(u + 2) = 5$, $(\frac{5}{q})_2 = -1$ ($q \equiv 2, 3 \bmod 5$ in all cases, and use Theorem C.3), and $(\frac{5}{q})_3 = -1$ as follows. $x \equiv 28, 100, 172 \bmod 360$ gives $(t, f) \equiv 5, 3 \bmod 6$, and $x \equiv 55, ..., 307 \bmod 360$ gives

$(t, f) \equiv (2, 0) \bmod 6$, so applying Equation (C.1)-(i) to both gives one of $(L, M) \equiv (1, 4), (3, 3), (4, 1) \bmod 5$, so that $LM \not\equiv 0 \bmod 5$. $x \equiv 43, 115, 259$ gives $(t, f) \equiv (2, 2) \bmod 6$, so using Equation (C.1)-(ii) further gives $(L, M) \equiv (1, 4), (4, 1) \bmod 5$, so that $LM \not\equiv 0 \bmod 5$. Finally, $x \equiv 139 \bmod 360$ gives $(t, f) \equiv (2, 4) \bmod 6$, so we use Equation (C.1)-(iii) to give $(L, M) \equiv (1, 1) \bmod 5$, implying $LM \not\equiv 0 \bmod 5$. Thus, $(\frac{5}{q})_3 = (\frac{5}{q})_2 = -1$ by Theorem C.4-(iii), so that $v^6 - (u + 2)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_3: $x \equiv 187, 283, 355 \bmod 360$ implies $q \equiv 7 \bmod 336$, $x \equiv 412, 772 \bmod 1800$ implies $q \equiv 343 \bmod 360$, $x \equiv 616, 976, 256 \bmod 1080$ implies $q \equiv 331 \bmod 360$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 1)$. Note that $2, 3 \mid q - 1$, and this time $N_{2,1}(u + 3) = 10$. For $x \equiv 187, 283, 355 \bmod 360$ and $x \equiv 412, 772 \bmod 1800$ we will prove that 2 is a quadratic residue but a cubic non-residue, whilst 5 is a quadratic non-residue but is a cube in $\mathbb{F}_q$. 2 being a quadratic residue follows from Prop. C.2. 5 being a quadratic non-residue follows from $q \equiv 2, 3 \bmod 5$ for these cases. For $x \equiv 187 \bmod 360$ and $x \equiv 283, 355 \bmod 360$, we have $(t, f) \equiv (2, 2) \bmod 6$ and $(t, f) \equiv (2, 4) \bmod 6$ respectively, which use Equation (C.1)-(ii) and Equation (C.1)-(iii) respectively to show that L and M are always odd, meaning that 2 is a cubic non-residue. Furthermore, both cases further reveal that $L \equiv 0 \bmod 5$ so that 5 is always a cubic residue. Combining $(\frac{2}{q})_2 = 1$, $(\frac{5}{q})_2 = -1$, $(\frac{2}{q})_3 = -1$ and $(\frac{5}{q})_3 = 1$ yields the result for $x \equiv 187, 283, 355 \bmod 360$ and $x \equiv 412, 772 \bmod 1800$. We now address $x \equiv 256, 616, 976 \bmod 1080$. This time we prove the opposite of the previous cases: namely that 5 is a quadratic but non-cubic residue, and that 2 is a non-quadratic but cubic residue. $(\frac{2}{q})_2 = -1$ follows from Prop. C.2. $(\frac{5}{q})_2 = 1$ follows from $q \equiv 1 \bmod 5$ and Theorem C.3. For all three congruences we have $(t, f) \equiv (5, 1) \bmod 6$ which invokes the use of Equation (C.1)-(iii) to show that L and M are always even (so that $(\frac{2}{q})_3 = 1$)), but $(L, M) \equiv (1, 2) \bmod 5$, so that $(\frac{5}{q})_3 = -1$ from $LM \not\equiv 0 \bmod 5$ and Theorem C.4-(iii). This completes the proof. □

T_4: $x \equiv 13, 61 \bmod 72$ implies $q \equiv 13 \bmod 24$, $x \equiv 70, 142, 214 \bmod 216$ implies $q \equiv 37 \bmod 72$, $x \equiv 118, ..., 1054 \bmod 1080$ implies $q \equiv 37 \bmod 72$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 2)$ (Prop. C.2). Since $2, 3 \mid q - 1$ and $N_{2,1}(u) = 2$, $(\frac{2}{q})_2 = -1$ (Prop. C.2), and $(\frac{2}{q})_3 = -1$ as follows. For $x \equiv 13 \bmod 72$, $t \equiv 2 \bmod 6$ and $f \equiv 4 \bmod 6$, so use Equation (C.1)-(iii) to see that L and M are both odd. For $x \equiv 61 \bmod 72$, $t \equiv f \equiv 2 \bmod 6$, so use Equation (C.1)-(ii) to see that L and M are both odd. For $x \equiv 70, 142, 214 \bmod 216$, $t \equiv f \equiv 5 \bmod 6$, so use Equation (C.1)-(ii) to further observe that L and M are again both odd. For all $x \equiv 118, ..., 1054 \bmod 1080$, $f = 3M$ so use Equation (C.1)-(i) and observe that M is always odd. Thus $N_{2,1}(u) = 2$ and $(\frac{2}{q})_3 = (\frac{2}{q})_2 = -1$ by Theorem C.4-(i), so that $v^6 - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_5: $x \equiv 37, 181 \bmod 216$ implies $q \equiv 37 \bmod 144$, $x \equiv 94, ..., 670 \bmod 1080$ implies $q \equiv$

133 mod 216, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2+2)$ (Prop. C.2). Since $2, 3 \mid q-1$ and $N_{2,1}(u) = 6$, which is not a quadratic or cubic residue as follows. To show $(\frac{6}{q})_2 = -1$, we see immediately that $(\frac{-2}{q})_2 = -1$ from Prop. C.2. To see that $(\frac{3}{q})_2 = 1$, we use Theorem C.3-(b) with $q \equiv 1 \bmod 12$. For the cubic non-residuosity, we have to split the cases. Observe that for $x \equiv 37, 181 \bmod 216$ we have $(t, f) \equiv (2, 0) \bmod 6$ so that Equation (C.1)-(i) can be used to see that $(L, M) \equiv (2, \pm 2) \bmod 3$, so that 3 is a cubic non-residue. On the other hand, $(L, M) \equiv (0, 0) \bmod 2$ for this case so that 2 is a cubic residue, which concludes the first case(s). For $x \equiv 94, ..., 670 \bmod 1080$, we always have $(t, f) \equiv (5, 1) \bmod 6$, so using Equation (C.1)-(iii) gives $(L, M) \equiv (4, \pm 2) \bmod 6$, so that 3 is again a cubic non-residue but a quadratic residue. Thus, $(\frac{6}{q})_3 = (\frac{6}{q})_2 = -1$ by Theorem C.4-(i),(ii), so that $v^6 - (u+2)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_6: $x \equiv 25, ..., 337 \bmod 360$ implies $q \equiv 1 \bmod 24$, $x \equiv 10, 82, 288 \bmod 360$ implies $q \equiv 73 \bmod 144$. In all cases, $q \equiv 2, 3 \bmod 5$ so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2+5)$. Now, $N_{2,1}(u) = 5$ so it remains to show $(\frac{5}{q})_3 = -1$. $x \equiv 73, 145, 217 \bmod 360$ gives $(t, f) \equiv (2, 0) \bmod 6$ so that Equation (C.1)-(i) gives $(L, M) \equiv (1, 4), (3, 3), (4, 1) \bmod 5$. For $x \equiv 25, 169, 313 \bmod 360$ we get $(t, f) \equiv (2, 2) \bmod 6$, so applying Equation (C.1)-(ii) gives $(L, M) \equiv (1, 4), (4, 1) \bmod 5$. $x \equiv 49, 337 \bmod 360$ gives $(t, f) \equiv (2, 4) \bmod 6$, so applying Equation (C.1)-(iii) gives $(L, M) \equiv (3, 2), (1, 1) \bmod 5$. Lastly, $x \equiv 10, 82, 288 \bmod 360$ all give $(t, f) \equiv (5, 3) \bmod 6$, so we can apply Equation (C.1)-(i) to see $(L, M) \equiv (3, 3), (1, 4) \bmod 5$. In all cases then, $LM \not\equiv 0 \bmod 5$, so that $(\frac{5}{q})_3 = (\frac{5}{q})_2 = -1$ by Theorem C.4-(iii), so $v^6 - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

b_1: $n \equiv 0 \bmod 12$. b must be square and cube, and one of $\{1, g, g^2, g^3, g^4, g^5\}$ for g non-square and non-cube, so $b = 1$ is the only option. □

b_2: $n \equiv 27 \bmod 108$, $q \equiv 1 \bmod 16$. $(t, f) \equiv (2, 0) \bmod 3$, so apply Algorithm C.2-Step 1 and take $b = 16$. $(\frac{2}{q})_2 = 1$ by Prop C.2. so $8 = \mu^6$ for $\mu^2 = 2$, thus the curve with $b = 16/\mu^6 = 2$ is isomorphic. □

b_{-2}: $n \equiv 27 \bmod 432$, $q \equiv 19 \bmod 72$. $(t, f) \equiv (2, 0) \bmod 3$, so apply Algorithm C.2 - Step 1 and take $b = 16$. This time $(\frac{-2}{q})_2 = 1$ by Prop C.2, so $-8 = \mu^6$ for $\mu^2 = -2$, thus the curve with $b = 16/\mu^6 = -2$ is isomorphic. □

b_4: $n \equiv 3 \bmod 36$, $q \equiv 1 \bmod 12$. Proof is identical to case $x_0 \equiv 16 \bmod 72$ for $k = 24$ BLS curves in [CLN11, Prop. 3]. □

b_3: $n \equiv 15 \bmod 24$, $q \equiv 1 \bmod 12$. There are three cases that arise: $(t, f) \equiv (2, 0), (2, 1), (2, 2) \bmod 3$. For $(t, f) \equiv (2, 0) \bmod 3$, we terminate with $b = 16$ from C.2, so $b = 3$ follows from observing $16/3$ (equivalently $2^4 3^5$) is μ^6 for some μ, which follows from the cubic and quadratic reciprocities of 2 and 3. For the other two cases $(t, f) \equiv (2, 1) \bmod 3$ and $(t, f) \equiv (2, 2) \bmod 3$, we use Alg. C.1 and take $b'^{(q-1)/6} = 12^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$ and $b'^{(q-1)/6} = 12^{(q-1)/6} \equiv 2t/(-3f - t) \bmod q$

respectively, so we can take $b = 16b'/2^6 = 3$ in both cases. □

b_{-3}: $n \equiv 147 \bmod 216$, $q \equiv 7 \bmod 12$. This time the two latter cases of the previous proof arise: $(t, f) \equiv (2, 1) \bmod 3$ and $(t, f) \equiv (2, 2) \bmod 3$, so again we use Alg. C.1, but this time it we take $b' = -12$ to see $b'^{(q-1)/6} = -12^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$ and $b'^{(q-1)/6} = -12^{(q-1)/6} \equiv 2t/(-3f - t) \bmod q$ respectively, so we can take $b = 16b'/2^6 = -3$ in both cases. □

b_{-5}: $n \equiv 3 \bmod 360$, $q \equiv 727 \bmod 1620$. We always have $(t, f) \equiv (2, 1) \bmod 3$, so Alg. C.1 with $b' = -20$ gives $b'^{(q-1)/6} = -20^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$, so we can take $b = 16b'/2^6 = -5$. □

b_5: $n \equiv 75 \bmod 900$, $q \equiv 214 \bmod 810$. Again we always have $(t, f) \equiv (2, 1) \bmod 3$, so Alg. C.1 this time with $b' = 20$ gives $b'^{(q-1)/6} = 20^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$, so we can take $b = 16b'/2^6 = 5$. □

b_9: $n \equiv 3 \bmod 12$, $q \equiv 1 \bmod 6$. Two cases: $(t, f) \equiv (2, 0) \bmod 3$ means $b = 16$ is the curve from Alg. C.1. It is easily seen that $(\frac{36}{q})_3 = 1$, so $(\frac{36}{q})_6 = 1$, meaning we can multiply b by $36/2^6$ to get the isomorphic curve with $b = 9$. For the second case we have $(t, f) \equiv (2, 1) \bmod 3$, so Alg. C.1 - Step 3 with $b' = 36$ gives $b'^{(q-1)/6} = 36^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$, so we can take $b = 16b'/64 = 9$. □

b_{10}: $n \equiv 183 \bmod 240$, $q \equiv 37 \bmod 120$. Two cases: $(t, f) \equiv (2, 0) \bmod 3$ means $b = 16$ is the curve from Alg. C.1. It is easily seen that $(\frac{40}{q})_6 = 1$, meaning we can multiply b by $40/2^6$ to get the isomorphic curve with $b = 10$. For the second case we have $(t, f) \equiv (2, 1) \bmod 3$, so Alg. C.1 - Step 3 with $b' = 40$ gives $b'^{(q-1)/6} = 40^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$, so we can take $b = 16b'/64 = 10$. □

$k = 16$ KSS curves

T_1: $x' \equiv 5, 37, 61, 93 \bmod 112$, $x' \equiv 47, 79 \bmod 112$, $x' \equiv 23, 103 \bmod 112$ all imply $q \equiv 5 \bmod 8$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 2)$ by Prop. C.2. Now, $N_{2,1}(u) = 2$ and we already have $(\frac{2}{q})_2 = -1$, so that $x^8 - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_2: $x' \equiv 19, ..., 1531 \bmod 1680$, $x' \equiv 1153, 1633 \bmod 1680$ all imply $q \equiv 17 \bmod 24$. Using Theorem C.3-(b), with $q \equiv 5 \bmod 12$, and since the odd squares modulo 12 are either 1 or 9, we have that $(\frac{\pm 3}{q})_2 = -1$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 3)$. We also have that $N_{2,1}(u) = 3$, so that $x^8 - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_3: This proof requires a special splitting of the elements in bunches. Namely, $x' \equiv 9, 89 \bmod 560$ implies $q \equiv 57 \bmod 80$; $x' \equiv 121, 201$ implies $q \equiv 73 \bmod 180$; $x' \equiv 401, 1601$ implies $q \equiv 193 \bmod 240$; $x' \equiv 929, 1409$ implies $q \equiv 97 \bmod 240$. We can now use Theorem C.3-(b), with $q \equiv 13, 17 \bmod 20$, and since the odd squares modulo 20 are either $1, 5$ or 9, we have that $(\frac{\pm 5}{q})_2 = -1$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 5)$.

We also have that $N_{2,1}(u) = 5$, so that $x^8 - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

a_1 : $n \equiv 2500 \bmod 10000$, $q \equiv 5 \bmod 8$. $(L, M) \equiv (1, 2) \bmod 4$, so we use Step 2 of Alg. C.1 and -1 is easily seen to be a square that is not a quartic residue. □

a_2 : $n \equiv 0 \bmod 1250$, $q \equiv 1 \bmod 4$. Two cases arise: $(L, M) \equiv (3, 0) \bmod 4$, so use Step 2 of C.1 where $(\frac{-2}{q})_2 = 1$ but $(\frac{-2}{q})_4 = -1$ gives the result. For $(L, M) \equiv (2, 3) \bmod 4$, we use Step 3 of Alg. C.1 and the fact that $-2^{(q-1)/4} \equiv L/M \bmod q$ to give the result. □

a_{-2} : $n \equiv 1250 \bmod 40000$, $q \equiv 13 \bmod 16$. $(L, M) \equiv (2, 3) \bmod 4$, and this time we have $2^{(q-1)/4} \equiv L/M \bmod q$. □

a_3 : $n \equiv 0 \bmod 1250$, $q \equiv 1 \bmod 8$. Two cases: $(L, M) \equiv (3, 0) \bmod 4$, so Step 2 of C.1 and $(\frac{-3}{q})_2 = 1$ but $(\frac{-3}{q})_4 = -1$ gives the result. For the second case, $(L, M) \equiv (0, 1) \bmod 4$, so Step 3 of Alg. C.1 and $-3^{(q-1)/4} \equiv L/M$ gives the result. □

a_5 : $n \equiv 0 \bmod 1250$, $q \equiv 1 \bmod 8$. Two cases: $(L, M) \equiv (3, 0) \bmod 4$ so Step 2 of Alg. C.1 with $(\frac{-5}{q})_2 = 1$ and $(\frac{-5}{q})_4 = -1$ gives the result. For $(L, M) \equiv (0, 1) \bmod 4$, Step 3 of Alg. C.1 with $-5^{(q-1)/4} \equiv L/M \bmod q$ finishes the proof. □

$k = 18$ KSS curves

T_1: $x' \equiv 4, 7, 16, 31 \bmod 36$ implies $q \equiv 1 \bmod 6$. We need to prove $(\frac{2}{q})_3 = -1$. $x' \equiv 4, 31 \bmod 36$ gives $t \equiv 1 \bmod 6$ and $f \equiv 3 \bmod 6$, so $f \equiv 3M$ and further $M \equiv 1 \bmod 2$, so we can use Equation (C.1)-(i) to give $4q = L^2 + 27M^2$, where L and M are both odd. $x' \equiv 7, 16 \bmod 36$ gives $t \equiv f \equiv 1 \bmod 6$, so we can use Equation (C.1)-(ii) to further show that L and M are both odd. Thus, $(\frac{\pm 2}{q})_3 = -1$ by Theorem C.4-(i), so that $\mathbb{F}_{q^3} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^3 + 2)$. Note that $x' \equiv 4, 16 \bmod 36$ gives $q \equiv 5 \bmod 8$, and $x' \equiv 7, 31 \bmod 36$ gives $q \equiv 7 \bmod 8$, so $(\frac{-2}{q})_2 = -1$ by Prop. C.2. Now, $N_{3,1}(u) = -2$ and $(\frac{2}{q})_2 = -1$, so that $x^6 - u$ is irreducible in $\mathbb{F}_{q^3}[x]$ by Theorem C.1. □

T_2: $x' \equiv 13, 25 \bmod 36$ implies $q \equiv 7 \bmod 24$. We need to prove that $(\frac{2}{q})_3 = -1$. With $x' \equiv 13 \bmod 36$, $f \equiv 0 \bmod 3$, i.e. $f = 3M$, insists use of Equation (C.1)-(i), which further reveals $4q = L^2 + 27M^2$ has L and M as odd. With $x' \equiv 25 \bmod 36$, $f \equiv t \equiv 1 \bmod 6$ insists use of Equation (C.1)-(ii) to give $(3f + t)/2$ and $(t - f)/6$ both odd. Thus, $\mathbb{F}_{q^3} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^3 + 2)$. This time, we have $N_{3,1}(2u) = -16$ and $(\frac{-16}{q})_2 = -1$ (since $(\frac{-1}{q})_2 = -1$ and $-16 = -1 \cdot 4^2$), and further $(\frac{-16}{q})_3 = -1$ (since $(\frac{-2}{q})_3 = -1$ by Theorem C.4-(i) and $-16 = -2 \cdot 2^3$), so $x^6 - u$ is irreducible in $\mathbb{F}_{q^3}[x]$ by Theorem C.1. □

T_3: $x' \equiv 1, 28, 37, 64 \bmod 108$ implies $q \equiv 7 \bmod 18$, and also that $f \equiv 2, 8 \bmod 9$, so that $(\frac{3}{q})_3 = -1$ by Theorem C.4-(ii), and $\mathbb{F}_{q^3} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^3 + 3)$. Now,

$N_{3,1}(u) = -24$, which is not a cubic residue (since -3 isn't). To apply Theorem C.1, it remains to show that $(\frac{-24}{q})_2 = -1$. $x' \equiv 1, 28, 37, 64 \bmod 108$ also implies $q \equiv 3, 5 \bmod 8$, so that $(\frac{2}{q})_2 = -1$. Since $-24 = 2 \cdot -3 \cdot 2^2$, and $(\frac{2}{q})_2 = -1$, we have that $(\frac{-24}{q})_2 \cdot (\frac{-3}{q})_2 = -1$, so it suffices to show that $(\frac{-3}{q})_2 = 1$. We have to split the possible congruences: for $x' \equiv 1, 37$ we always have $q \equiv 7 \bmod 12$, and taking $q = 3$ in Theorem C.3 does the trick, since 1 and 9 are the only "odd squares" modulo 12. Thus, for $x' \equiv 1, 37$, $(\frac{3}{q})_2 = -1$ and $(\frac{-1}{q})_2 = -1$ gives $(\frac{-3}{q})_2 = 1$. For $x' \equiv 28, 64$, we have $q \equiv 1 \bmod 12$, which does just the opposite, meaning $(\frac{3}{q})_2 = 1$, but $(\frac{-1}{q})_2 = 1$ also, meaning $(\frac{-3}{q})_2 = 1$ as well. □

T_4: $x' \equiv 22, 58, 142, 178 \bmod 180$ implies $q \equiv 1 \bmod 12$. To prove $(\frac{-2}{q})_3 = -1$, we need to split into two separate cases and use Theorem C.4-(i). For $x' \equiv 22, 58 \bmod 180$, we have $f \equiv 0 \bmod 3$, i.e. $f = 3M$, insists use of Equation (C.1)-(i), which further reveals $4q = L^2 + 27M^2$ has L and M always odd. For $x' \equiv 142, 178 \bmod 180$ we have $t \equiv f \equiv 1 \bmod 6$ and application of Equation (C.1)-(ii) shows that L and M are both odd. Thus, $\mathbb{F}_{q^3} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^3 + 2)$. $N_{3,1}(5u) = -250$, and $(\frac{-250}{q})_3 = -1$ follows from $(\frac{2}{q})_3 = -1$ (since $-250 = -2 \cdot 5^3$), so it remains to prove $(\frac{-250}{q})_2 = -1$ before applying Theorem C.1. Since $q \equiv 1 \bmod 4$, $(\frac{-1}{q})_2 = 1$ so that $(\frac{-250}{q})_2 = (\frac{10}{q})_2$. Further, $x' \equiv 22, 58, 142, 178 \bmod 180$ implies $q \equiv 1 \bmod 8$ so Prop. C.2 says that $(\frac{2}{q})_2 = 1$, meaning that $(\frac{10}{q})_2 = (\frac{2}{q})_2 \cdot (\frac{5}{q})_2 = (\frac{5}{q})_2$. For this, combine the fact that $x' \equiv 22, 58, 142, 178 \bmod 180$ implies $q \equiv 2, 3 \bmod 5$ with Theorem C.3-(a) to give that $(\frac{5}{q})_2 = -1$. Thus, $(\frac{-250}{q})_2 = -1$ so that $x^6 - 2u$ is irreducible in $\mathbb{F}_{q^3}[x]$ by Theorem C.1. □

T_5: $x' \equiv 19, 181, 208, 262 \bmod 270$ implies $q \equiv 7 \bmod 54$. We now show that $(\frac{-5}{q})_3 = (\frac{5}{q}) - 1$ using Theorem C.4 - (iii). First, for $x' \equiv 19, 181, 208, 262 \bmod 270$, we always have $t \equiv 1 \bmod 6$ and $f \equiv 5 \bmod 6$, so we make use Equation (C.1)-(iii) and see that niether L nor M is divisible by 5. Thus, $(\frac{5}{q})_3 = -1$ and $\mathbb{F}_{q^3} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^3 + 5)$. $N_{3,1}(u) = -5$, so to finish the proof we need to show that $(\frac{-5}{q})_2 = -1$. We split the congruences into two cases: $x' \equiv 19, 181 \bmod 270$ gives $q \equiv 3 \bmod 4$ and $q \equiv \pm 1 \bmod 5$ which means firstly that $(\frac{-5}{q})_2 = -(\frac{5}{q})_2$, and also that $(\frac{5}{q})_2 = 1$ from Theorem C.3-(a). For the other two congruences $x' \equiv 208, 262 \bmod 270$, $q \equiv 1 \bmod 4$ and $q \equiv \pm 2 \bmod 5$ which means firstly that this time $(\frac{-5}{q})_2 = (\frac{5}{q})_2$, but secondly that $(\frac{5}{q})_2 = -1$ from Theorem C.3-(a). In both cases then, $(\frac{-5}{q}) = -1$ and $(\frac{-5}{q})_2 =) \frac{-5}{q})_3 = -1$, so that $x^6 - u$ is irreducible in $\mathbb{F}_{q^3}[x]$ by Theorem C.1. □

b_3: $n \equiv 16807 \bmod 37044$, $q \equiv 7 \bmod 36$. Two cases arise: $(t, f) \equiv (1, 1) \bmod 3$, so Step 4 of Alg. C.2 with $b' = 12$ gives $12^{(q-1)/6} \equiv 2t/(3f + t) \bmod q$, and dividing $b = 16b'$ by 2^6 gives the result. For the other case, $(t, f) \equiv (1, 2) \bmod 3$, so Step 3 of Alg. C.2 with $b' = 12$ (and the division by 2^6) gives the same result. □

b_{-9}: $n \equiv 4459 \bmod 37044$. We always have the case $(t, f) \equiv (1, 2) \bmod 3$, so Step 4 of

Alg. C.2 with $b' = -36$ gives $-36^{(q-1)/6} \equiv 2t/(-3f+t) \bmod q$. Division of $b = 16b'$ by 2^6 gives the result. □

b_5: $n \equiv 343 \bmod 2058$, $q \equiv 1 \bmod 6$. Two cases arise: $(t, f) \equiv (1, 0) \bmod 3$, so Step 2 of Alg. C.2 with $b' = 20$ gives $20^{(q-1)/6} \equiv -1 \bmod q$ gives the result. For the second case, $(t, f) \equiv (1, 2) \bmod 3$ so Step 4 of Alg. C.2 with $b' = 20$ gives the same constant. □

b_7: $n \equiv 343 \bmod 2058$, $q \equiv 1 \bmod 6$. Three cases arise: $(t, f) \equiv (1, 0) \bmod 3$ means Step 2 of Alg. C.2 applies, here with $b' = 36$ gives $36^{(q-1)/6} \equiv -1 \bmod q$. The second two cases are $(t, f) \equiv (1, 1) \bmod 3$ and $(t, f) \equiv (1, 2) \bmod 3$, which both use Step 4. of Alg. C.2 and $b' = 36$ to give $36^{(q-1)/6} \equiv 2t/(3f+t), 2t/(-3f+t) \bmod q$ respectively. All three cases give $b = 16b'$ which can be divided by 2^6 to give $b = 7$. □

b_{-7}: $n \equiv 53851 \bmod 86436$, $q \equiv 115 \bmod 252$. One case: $(t, f) \equiv (1, 2) \bmod 3$ so Step 4. of Alg C.2 with $b' = -28$ gives $-28^{(q-1)/6} \equiv 2t/(-3t+f) \bmod q$. Division of $b = 16b'$ by 2^6 gives the result. □

b_6: $n \equiv 22981 \bmod 24696$, $q \equiv 61 \bmod 72$. Two cases arise, both requiring Step 4 of Alg. C.2. Namely $(t, f) \equiv (1, 2) \bmod 3$ and $(t, f) \equiv (1, 1) \bmod 3$ take $b' = 24$ to give $24^{(q-1)/6} \bmod q$ as $2t/(-3f+t)$ and $2t/(3f+t)$ respectively. Division of $b = 16b'$ by 2^6 gives the result. □

b_2: $n \equiv 12691 \bmod 18522$, $q \equiv 31 \bmod 54$. $(t, f) \equiv (1, 0) \bmod 3$ is the only case, so taking $b' = 8$ gives $8^{(q-1)/6} \equiv -1 \bmod q$ in Step 2 of Alg. C.2, and dividing $b = 16b'$ by 2^6 gives the result. □

b_{-4}: $n \equiv 4459 \bmod 12348$, $q \equiv 7 \bmod 36$. The only case is $(t, f) \equiv (1, 1) \bmod 3$ which requires Step 4 of Alg. C.2 with $b' = -16$ to give $-16^{(q-1)/6} \equiv 2t/(3f+t)$ to give the result (again, after division of b by 2^6). □

b_{-2}: $n \equiv 49735 \bmod 74088$, $q \equiv 31 \bmod 216$. The only case is $(t, f) \equiv (1, 0) \bmod 3$, for which we can use Step 2 of Alg. C.2 to deduce that $b' = -8$ always gives $-8^{(q-1)/6} \equiv -1 \bmod q$. Division of b by 2^6 gives $b = -2$. □

b_{10}: $n \equiv 10633 \bmod 41160$, $q \equiv 97 \bmod 120$. Two cases arise: $(t, f) \equiv (1, 0) \bmod 3$ requires Step 2 of Alg. C.2 with $b' = 40$ to always give $40^{(q-1)/6} \equiv -1 \bmod q$. The second case is $(t, f) \equiv (1, 1) \bmod 3$, which uses $b' = 40$ in Step 4 of Alg. C.2 to give $40^{(q-1)/6} \equiv 2t/(3f+t) \bmod q$. In both cases we again divide b by 2^6 to give the smaller constant. □

$k = 27$ BLS curves

T_1: $x \equiv 2 \bmod 9$ implies $q \equiv 7 \bmod 9$. Once case: $t \equiv 5 \bmod 6$ and $f \equiv 1 \bmod 6$, so applying Equation (C.1)-(iii) gives further that $M \not\equiv 0 \bmod 3$ so Theorem C.4-(ii) gives $(\frac{3}{q})_3 = -1$. Thus, $\mathbb{F}_{q^3} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^3+3)$, and furthermore since $N_{3,1}(u) = -3$, we immediately have that $x^9 - u$ is irreducible in $\mathbb{F}_{q^3}[x]$ by Theorem C.1. □

T_2: $x \equiv 8 \bmod 45$ implies $q \equiv 37 \bmod 45$. Again, $x \equiv 8 \bmod 45$ gives $t \equiv 5 \bmod 6$ and $f \equiv 1 \bmod 6$, insisting the use of Equation (C.1)-(iii) which gives both $L, M \not\equiv 0 \bmod 5$, so $(\frac{5}{q})_3 = -1$ by Theorem C.4-(iii). Thus, $\mathbb{F}_{q^3} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^3 + 5)$, and since $N_{3,1}(u) = -5$, we immediately have that $x^9 - u$ is irreducible in $\mathbb{F}_{q^3}[x]$ by Theorem C.1. □

T_3: $x \equiv 17, ..., 269 \bmod 315$ implies $q \equiv 1 \bmod 45$. Again, $x \equiv 17, ..., 269 \bmod 45$ gives $t \equiv 5 \bmod 6$ and $f \equiv 1 \bmod 6$, so applying Equation (C.1)-(iii) to see that $L, M \not\equiv 0 \bmod 7$ and Theorem C.4-(iv) gives $(\frac{7}{q})_3 = -1$. Thus, $\mathbb{F}_{q^3} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^3 + 7)$, and since $N_{3,1}(u) = -7$, $x^9 - u$ is irreducible in $\mathbb{F}_{q^3}[x]$ by Theorem C.1. □

b_{-5}: $n \equiv 1083 \bmod 1350$. We always have $(t, f) \equiv (2, 1) \bmod 3$, so Step 3 of Alg. C.2 with $b' = -20$ gives $-20^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$. Division by 2^6 gives a smaller constant as usual. □

Other b's: All other proofs are identical, i.e. have $(t, f) \equiv (1, 2) \bmod 3$ and use Step 3 of Alg. C.2 with the appropriate b'. □

$k = 32$ KSS curves

T_1: $x' \equiv 453, ..., 2893 \bmod 3824$, $x' \equiv 1887, 2415 \bmod 3824$ and $x' \equiv 503, 3799 \bmod 3824$ all imply $q \equiv 5 \bmod 8$, so that $(\frac{2}{q})_2 = (\frac{-2}{q})_2 - 1$ by Prop. C.2. Thus, $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 2)$, and since $N_{2,1}(u) = 2$, $x^{16} - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_2: $x' \equiv 7145, 7673 \bmod 11472$ implies $q \equiv 17 \bmod 48$, $x' \equiv 2843, 3371, 8579, 9148 \bmod 11472$ implies $q \equiv 17 \bmod 24$. So we always have $q \equiv 5 \bmod 12$. The "odd squares" modulo 12 are 1 and 9 only, so that Theorem C.3-(ii) allows us to immediately conclude that $(\frac{3}{q})_2 = (\frac{-3}{q})_2 = -1$ in all cases. Thus, $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 3)$, and since $N_{2,1}(u) = 3$, $x^{16} - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

a_1: $n \equiv 81573072100 \bmod 117465223824$, $q \equiv 5 \bmod 8$. $(L, M) \equiv (1, 2) \bmod 4$, so using Step 2 of Alg. C.1 with $(\frac{-1}{q})_2 = 1$ and $(\frac{-1}{q})_4 = -1$ gives the result. □

a_{-2}: $n \equiv 8157307210 \bmod 939721790592$, $q \equiv 5 \bmod 16$. $(L, M) \equiv (2, 1) \bmod 4$, so using Step 3 of Alg. C.1 with $2^{(q-1)/4} \equiv L/M \bmod q$ gives the result. □

a_2: $n \equiv 8157307210 \bmod 14683152978$, $q \equiv 1 \bmod 4$. Two cases: $(L, M) \equiv (2, 1) \bmod 4$, so using Step 3 of Alg. C.1 with $-2^{(q-1)/4} \equiv L/M \bmod q$ gives the first result. For the second result $(L, M) \equiv (3, 0) \bmod 4$ so Step 2 of Alg. C.1 with $-2^{(q-1)/4} \equiv -1 \bmod q$ completes the proof. □

a_3: $n \equiv 301820366770 \bmod 469860895296$, $q \equiv 17 \bmod 72$. $(L, M) \equiv (0, 3) \bmod 4$ is the only scenario, so Step 2 of Alg. C.1 with $-3^{(q-1)/4} \equiv -1 \bmod q$ gives the result. □

$k = 36$ KSS curves

T_1: $x' \equiv 1880, ..., 2264 \bmod 2664$ implies $q \equiv 19 \bmod 24$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2+1)$. Now, $N_{2,1}(u+1) = 2$, and $(\frac{2}{q})_2 = -1$ from Prop. C.2. To prove $(\frac{2}{q})_3 = -1$, we need to split the congruences into 4 sets. Firstly, $x' \equiv 1376, 1880 \bmod 2664$ gives $t \equiv 1 \bmod 6$ and $f \equiv 3 \bmod 6$, so Equation (C.1)-(i) with $f = 3M$ gives L and M both odd. For $x' \equiv 821, 1325 \bmod 2664$ gives $t \equiv 4 \bmod 6$ and $f \equiv 2 \bmod 6$, so Equation (C.1)-(iii) reveals that L and M are both odd. For $x' \equiv 437, 2597 \bmod 2664$, we have $t \equiv f \equiv 4 \bmod 6$, and using Equation (C.1)-(ii) reveals that L and M are both odd. Lastly, $x' \equiv 104, 2264 \bmod 2664$ gives $t \equiv f \equiv 1 \bmod 6$, so again using Equation (C.1)-(ii) gives L and M as both odd. Thus, $(\frac{2}{q})_3 = -1$ by Theorem C.4-(i) so that $x^{18} - (u+1)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_2: $x' \equiv 3152, ..., 7652 \bmod 13320$ implies $q \equiv 31 \bmod 36$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2+1)$. $N_{2,1}(u+2) = 5$, and $(\frac{5}{q})_2 = -1$ since $x' \equiv 3152, ..., 7652 \bmod 13320$ always gives $q \equiv 2, 3 \bmod 5$, allowing us to apply Theorem C.3-(i). To prove $(\frac{5}{q})_3 = -1$, we must split the congruences into 6 different sets: $x' \equiv 932, 6260, 4100, 12092 \bmod 13320$ gives $t \equiv f \equiv 1 \bmod 6$, so using Equation (C.1)-(ii) gives $(L, M) \equiv (1, 4), (2, 2) \bmod 5$. $x' \equiv 3152, ..., 7652 \bmod 13320$ gives $t \equiv 1 \bmod 6$ and $f \equiv 5 \bmod 6$ so using Equation (C.1)-(iii) gives $(L, M) \equiv (3, 3), (1, 4) \bmod 5$. $x' \equiv 44, ..., 11204 \bmod 13320$ gives $t \equiv 1 \bmod 6$ and $f \equiv 3 \bmod 6$ so using Equation (C.1)-(i) gives $(L, M) \equiv (1, 1), (1, 4) \bmod 5$. $x' \equiv 1709, ...12869 \bmod 13320$ gives $t \equiv 4 \bmod 6$ and $f \equiv 0 \bmod 6$, so using Equation (C.1)-(i) with $f \equiv 3M$ gives $(L, M) \equiv (1, 1), (1, 4) \bmod 5$. $x' \equiv 1265, ..., 12425 \bmod 13320$ gives $t \equiv f \equiv 4 \bmod 6$ so we can use Equation (C.1)-(ii) to further give $(L, M) \equiv (1, 4), (2, 2) \bmod 5$. Lastly, $x' \equiv 2657, ..., 10649 \bmod 13320$ gives $t \equiv 4 \bmod 6$ and $f \equiv 2 \bmod 6$, and then Equation (C.1)-(iii) gives $(L, M) \equiv (3, 3), (1, 4) \bmod 5$. Thus, $(\frac{5}{q})_3 = (\frac{5}{q})_2 = -1$ by Theorem C.4-(iii), so that $x^{18} - (u+2)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_3: $x' \equiv 5372, ..., 10145 \bmod 13320$ implies $q \equiv 7 \bmod 12$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2+1)$. $N_{2,1}(u+3) = 10$. To show $(\frac{10}{q})_3 = (\frac{10}{q})_2 = -1$ we must split the congruences. $x' \equiv 3929, ..., 10145 \bmod 13320$ implies $q \equiv 7 \bmod 24$ so that $(\frac{2}{q})_2 = 1$, and also that $q \equiv 2, 3 \bmod 5$ so that $(\frac{5}{q})_2 = -1$, which gives $(\frac{10}{q})_2 = -1$. Each of the four congruences give a different pair for $(t, f) \bmod 6$: $x' \equiv 5372 \bmod 13320 \rightarrow (t, f) \equiv (1, 3) \bmod 6$, so using Equation (C.1)-(i) gives L, M both odd but $L \equiv 0 \bmod 5$. $x' \equiv 3929 \bmod 13320 \rightarrow (t, f) \equiv (4, 4) \bmod 6$, so using Equation (C.1)-(ii) gives L, M both odd but again $L \equiv 0 \bmod 5$. $x' \equiv 8924 \bmod 13320 \rightarrow (t, f) \equiv (1, 1) \bmod 6$, so using Equation (C.1)-(ii) again gives L, M both odd and $L \equiv 0 \bmod 5$. $x' \equiv 10145 \bmod 13320 \rightarrow (t, f) \equiv (4, 2) \bmod 6$, so using Equation (C.1)-(iii) this time gives L, M both odd and $L \equiv 0 \bmod 5$. Thus, for all four cases $(\frac{2}{q})_3 = -1$ by Theorem

C.4-(i) and $(\frac{5}{q})_3 = 1$ by Theorem C.4-(iii) so that $(\frac{10}{q})_3 = -1$. For the second set $x' \equiv 488, 4373, 5816 \bmod 13320$. For both $x' \equiv 488, 5816 \bmod 13320$, $(t, f) \equiv (1, 5) \bmod 6$ so using Equation (C.1)-(iii) gives both L and M as even, but with either $(L, M) \equiv (3, 1), (3, 4) \bmod 5$ so that $(\frac{2}{q})_3 = -1$ but $(\frac{5}{q})_3 = 1$ from C.4-(i) and (iii), meaning $(\frac{10}{q})_3 = -1$. Lastly, $x' \equiv 4373 \bmod 13320$ gives $(t, f) \equiv (4, 0) \bmod 6$ so Equation (C.1)-(i) shows that $(L, M) \equiv (2, 4) \bmod 10$, meaning again that $(\frac{10}{q})_3 = -1$. Thus, $(\frac{10}{q})_2 = (\frac{10}{q})_3 = -1$ in all cases so $x^{18} - (u + 3)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_4: $x' \equiv 710, ..., 2102 \bmod 2664$ implies $q \equiv 13 \bmod 24$ so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 2)$ (by Prop. C.2). $N_{2,1}(u) = 2$, and $(\frac{2}{q})_3 = -1$ as follows. Again, we need to split the possibilities: $x' \equiv 710, 1214 \bmod 2664$ gives $(t, f) \equiv (1, 3) \bmod 6$ so using Equation (C.1)-(i) gives L and M both odd. $x' \equiv 155, 659 \bmod 2664$ gives $(t, f) \equiv (4, 2) \bmod 6$ so that Equation (C.1)-(iii) gives both L and M as odd. $x' \equiv 1931, 2435 \bmod 2664$ gives $(t, f) \equiv (4, 4) \bmod 6$ so that this time Equation (C.1)-(ii) gives both L and M as odd. Lastly, $1598, 2102 \bmod 2664$ gives $(t, f) \equiv (1, 1) \bmod 6$ so again Equation (C.1)-(ii) gives both L and M as odd. Thus, $(\frac{2}{q})_3 = (\frac{2}{q})_3 = -1$ by Theorem C.4-(i), so that $x^{18} - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_5: $x' \equiv 9035, ..., 5210 \bmod 13320$ implies $q \equiv 37 \bmod 180$, and the only possibilities for q modulo 5 are $2, 3$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 5)$ by Theorem C.3. $N_{2,1}(u) = 5$, and $(\frac{5}{q})_3 = -1$ as follows. Again we require splitting the congruences: for $x' \equiv 4322, ..., 10982 \bmod 13320$ we have $(t, f) \equiv (1, 5) \bmod 6$ and $x' \equiv 1487 \bmod 13320$ we have $(t, f) \equiv (4, 2) \bmod 6$, so for both these cases Equation (C.1)-(iii) reveals that $(L, M) \equiv (3, 3) \bmod 5$ so that $LM \not\equiv 0 \bmod 5$. $x' \equiv 7874, 10034 \bmod 13320$ gives $(t, f) \equiv (1, 3) \bmod 6$ and $x' \equiv 6875, 11699, 9035 \bmod 13320$ gives $(t, f) \equiv (4, 0) \bmod 6$ so applying Equation (C.1)-(i) to both gives $(L, M) \equiv (1, 4), (1, 1) \bmod 5$ so that $LM \not\equiv 0 \bmod 5$. Lastly, $x' \equiv 770, 2930 \bmod 13320$ gives $(t, f) \equiv (1, 1) \bmod 6$ demanding the use of Equation (C.1)-(ii) to show that $(L, M) \equiv (1, 4) \bmod 5$ so that $LM \not\equiv 0 \bmod 5$. In all cases then, $(\frac{5}{q})_3 = (\frac{5}{q})_2 = -1$ by Theorem C.4-(iii), $x^{18} - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

b_5: $n \equiv 117649 \bmod 352947$, $q \equiv 1 \bmod 6$. Three cases arise: $(t, f) \equiv (1, 0) \bmod 3$ uses Step 2 of Alg. C.2 with $b' = 20$ to give $20^{(q-1)/6} \equiv -1 \bmod q$. $(t, f) \equiv (1, 1) \bmod 3$ needs Step 4 and $(t, f) \equiv (1, 2) \bmod 3$ both use Step 4 with $b' = 20$ to give $20^{(q-1)/6} \bmod q$ as $2t/(3f + t)$ and $2t/(-3f + t)$ respectively. All three cases require further division of b by 2^6 to give the smaller constant $b = 5$. □

b_2: $n \equiv 470596 \bmod 3176523$, $q \equiv 19 \bmod 54$. $(t, f) \equiv (1, 0) \bmod 3$ always, so Step 2 of Alg. C.2 with $b' = 8$ gives $8^{(q-1)/6} \equiv -1 \bmod q$, and division of b by 2^6 gives the result.

b_{10}: $n \equiv 117649 \bmod 1764735$, $q \equiv 1 \bmod 12$. Three cases arise: $(t, f) \equiv (1, 0) \bmod$

3 uses Step 2 of Alg. C.2 with $b' = 40$ to give $40^{(q-1)/6} \equiv -1 \bmod q$. $(t, f) \equiv (1, 1) \bmod 3$ needs Step 4 and $(t, f) \equiv (1, 2) \bmod 3$ both use Step 4 with $b' = 40$ to give $40^{(q-1)/6} \bmod q$ as $2t/(3f + t)$ and $2t/(-3f + t)$ respectively. All three cases require further division of b by 2^6 to give the smaller constant $b = 10$. □

b_{-1}: $n \equiv 470596 \bmod 2823576$, $q \equiv 7 \bmod 12$. Three cases arise: $(t, f) \equiv (1, 0) \bmod 3$ uses Step 2 of Alg. C.2 with $b' = -4$ to give $-4^{(q-1)/6} \equiv -1 \bmod q$. $(t, f) \equiv (1, 1) \bmod 3$ needs Step 4 and $(t, f) \equiv (1, 2) \bmod 3$ both use Step 4 with $b' = -4$ to give $-4^{(q-1)/6} \bmod q$ as $2t/(3f + t)$ and $2t/(-3f + t)$ respectively. All three cases require further division of b by 2^6 to give the smaller constant $b = -1$. □

b_{-4}: $n \equiv 30471091 \bmod 33882912$, $q \equiv 31 \bmod 36$. $(t, f) \equiv (1, 1) \bmod 3$ is the only case. Thus, $b' = -16$ into Step 4 of Alg. C.2 gives $-16^{(q-1)/6} \equiv 2t/(3f + t)$, and division of b by 2^6 gives $b = -4$. □

b_3: $n \equiv 30471091 \bmod 33882912$, $q \equiv 103 \bmod 108$. Two cases arise: $(t, f) \equiv (1, 1) \bmod 3$ and $(t, f) \equiv (1, 2) \bmod 3$, so applying Step 4 of Alg. C.2 with $b' = 12$ to both gives $-16^{(q-1)/6} \bmod q$ as $2t/(3f + t)$ and $2t/(-3f + t)$ respectively. Division by 2^6 gives the result. □

b_{-2}: $n \equiv 41765395 \bmod 101648736$, $q \equiv 127 \bmod 216$. We always have $(t, f) \equiv (1, 0) \bmod 3$, so Step 2 of Alg. C.2 with $b' = -8$ gives $-8^{(q-1)/6} \equiv -1 \bmod q$. Further division of $b = 16b'$ by 2^6 gives the result. □

b_{-5}: $n \equiv 166002739 \bmod 169414560$, $q \equiv 139 \bmod 180$. We always have $(t, f) \equiv (1, 2) \bmod 3$ so Step 4 of Alg. C.2 with $b' = -20$ gives $-20^{(q-1)/6} \equiv 2t/(-3f+t) \bmod q$. Again, further division of $b = 16b'$ by 2^6 gives the result. □

$k = 48$ BLS curves

T_1: $x \equiv 7, 31 \bmod 72$ implies $q \equiv 19 \bmod 24$, $x \equiv 16, 64 \bmod 72$ implies $q \equiv 19 \bmod 24$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 1)$. $N_{2,1}(u) = 2$, $(\frac{2}{q})_2 = -1$ (Prop. C.2) and $(\frac{2}{q})_3 = -1$ as follows. We have to prove each case separately: $x \equiv 7 \bmod 72$ gives $t \equiv f \equiv 2 \bmod 6$, whilst $x \equiv 16 \bmod 72$ gives $t \equiv f \equiv 5 \bmod 6$, so using Equation (C.1)-(ii) gives L and M both odd for both cases. $x \equiv 31 \bmod 72$ gives $t \equiv 2 \bmod 6$ and $f \equiv 4 \bmod 6$, so using C.1-(iii) gives L and M both odd for both cases. Lastly, $x \equiv 64 \bmod 72$ gives $t \equiv 5 \bmod 6$ and $f \equiv 3 \bmod 6$, so applying Equation (C.1)-(i) further gives L and M both odd for both cases. Thus, $(\frac{2}{q})_3 = (\frac{2}{q})_2 = -1$ by Theorem C.4-(i), so that $x^{24} - (u + 1)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_2: $x \equiv 235, ..., 139 \bmod 360$ implies $q \equiv 7 \bmod 60$, $x \equiv 4, ..., 340 \bmod 360$ implies $q \equiv 7 \bmod 60$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 1)$. $N_{2,1}(u + 2) = 5$, which is not a quadratic residue since $x \equiv 235, ..., 139 \bmod 360$ gives $q \equiv 2 \bmod 5$, invoking Theorem C.3. To prove $(\frac{5}{q})_3 = -1$, we need to case bash. $x \equiv 55, 235 \bmod 360$ gives

$(t, f) \equiv (2, 0) \bmod 6$ so we can apply Equation (C.1)-(i) to further yield $(L, M) \equiv (1, 4) \bmod 5$, so that $LM \not\equiv 0 \bmod 5$, $x \equiv 115, 259 \bmod 360$ gives $(t, f) \equiv (2, 2) \bmod 6$ so we apply Equation (C.1)-(ii) to further yield $(L, M) \equiv (1, 1), (1, 4) \bmod 5$, giving $LM \not\equiv 0 \bmod 5$. Lastly, $x \equiv 139 \bmod 360$ gives $(t, f) \equiv (2, 4) \bmod 6$, so applying C.1-(iii) to further yield $(L, M) \equiv (1, 1) \bmod 5$ gives $LM \not\equiv 0 \bmod 5$. Thus, $(\frac{5}{q})_3 = (\frac{5}{q})_2 = -1$ by Theorem C.4-(iii), meaning that $x^{24} - (u+2)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_3: $x \equiv 13, 61 \bmod 72$ implies $q \equiv 13 \bmod 24$, $x \equiv 10, 34 \bmod 72$ implies $q \equiv 13 \bmod 24$,so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 2)$ from (Prop. C.2). $N_{2,1}(u) = 2$, and $(\frac{2}{q})_3 = -1$ as follows. $x \equiv 34, 61 \bmod 72$ gives $(t, f) \equiv (5, 5) \bmod 6$ and $(t, f) \equiv (2, 2) \bmod 6$ respectively, which insists use of Equation (C.1)-(ii) to give L and M as both odd. $x \equiv 10 \bmod 72$ gives $(t, f) \equiv (1, 0) \bmod 6$ so that Equation (C.1)-(i) can be used to show L is odd. Lastly, $x \equiv 13 \bmod 72$ gives $(t, f) \equiv (2, 4)$ so that C.1-(iii) can be used to show L and M are both odd. Thus, $(\frac{2}{q})_3 = (\frac{2}{q})_2 = -1$ by Theorem C.4-(i), so that $x^{24} - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_4: $x \equiv 37, 181 \bmod 216$ implies $q \equiv 37 \bmod 144$, $x \equiv 130, 202 \bmod 216$ implies $q \equiv 133 \bmod 216$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2 + 2)$. $N_{2,1}(u + 2) = 6$. We first have that $(\frac{2}{q})_2 = -1$ Prop. C.2, but $(\frac{2}{q})_3 = 1$ for all cases as follows. $x \equiv 37, 181 \bmod 216$ gives $(t, f) \equiv (2, 0) \bmod 6$ so that we can use Equation (C.1)-(i) to show that L and M are both even. $x \equiv 130, 202 \bmod 216$ gives $(t, f) \equiv (5, 1) \bmod 6$ so we can use C.1-(iii) to show that L and M are both even. Thus, $(\frac{2}{q})_3 = 1$. On the other hand, we show that $(\frac{3}{q})_2 = 1$ but $(\frac{3}{q})_3 = -1$. Note that $q \equiv 1 \bmod 12$ so that Theorem C.3-(b) gives $(\frac{3}{q})_2 = 1$. To show $(\frac{3}{q})_3 = -1$, the same congruences and corresponding (t, f) pairs immediately give that $M \not\equiv 0 \bmod 3$ in all cases. Thus, $(\frac{6}{q})_3 = (\frac{6}{q})_2 = -1$ by Theorem C.4-(i) and (ii), so that $x^{24} - (u + 2)$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

T_5: $x \equiv 25, 145, 49, 169 \bmod 360$ implies $q \equiv 97 \bmod 120$, $x \equiv 70, 190, 94, 214 \bmod 360$ implies $q \equiv 97 \bmod 120$, so that $\mathbb{F}_{q^2} = \mathbb{F}_q(u) = \mathbb{F}_q[u]/(u^2+5)$. $N_{2,1}(u) = 5$, and $(\frac{5}{q})_2 = -1$ (by Theorem C.3-(a) with $q \equiv 2 \bmod 5$), and further $(\frac{5}{q})_3 = -1$ as follows. $x \equiv 190 \bmod 360$ gives $(t, f) \equiv (5, 3) \bmod 6$ and $x \equiv 145 \bmod 360$ gives $(t, f) \equiv (2, 0) \bmod 6$. In both cases, Equation (C.1)-(i) gives $(L, M) \equiv (1, 4) \bmod 5$ so that $LM \not\equiv 0 \bmod 5$. For $x \equiv 25, 169 \bmod 360$, $(t, f) \equiv (2, 2) \bmod 6$ whilst for $x \equiv 70, 214 \bmod 360$, $(t, f) \equiv (5, 5) \bmod 6$, so Equation (C.1)-(ii) gives $(L, M) \equiv (1, 4), (4, 4) \bmod 5$ so that $LM \not\equiv 0 \bmod 5$. Lastly, $x \equiv 49 \bmod 360$ gives $(t, f) \equiv (2, 4) \bmod 6$ so application of C.1-(iii) further reveals that $(L, M) \equiv (1, 1) \bmod 5$, meaning that $LM \not\equiv 0 \bmod 5$. Thus, $(\frac{5}{q})_3 = (\frac{5}{q})_2 = -1$ by Theorem C.4-(iii), so that $x^{24} - u$ is irreducible in $\mathbb{F}_{q^2}[x]$ by Theorem C.1. □

b_1: $n \equiv 0 \bmod 12$. $n \equiv 0 \bmod 12$ needs b as square and cube, so for any non-square, non-cube g, 1 is the only possibility in $\{1, g, g^2, g^3, g^4, g^5\}$. □

b_{-2}: $n \equiv 27 \bmod 432$, $q \equiv 19 \bmod 72$. $(t, f) \equiv (2, 0) \bmod 3$ means $b = 16$, and $(\frac{-2}{q})_2 = 1$ so $-8 \equiv \mu^6$ for $\mu = \sqrt{-2}$, so $b = -2$ gives an isomorphic curve. □

b_{-3}: $n \equiv 147 \bmod 216$, $q \equiv 7 \bmod 12$. Two cases: $(t, f) \equiv (2, 1) \bmod 3$ and $(t, f) \equiv (2, 2) \bmod 3$, both of which use Step 3 of Alg. C.2 with $b' = -12$ to give $-12^{(q-1)/6} \bmod q$ as $2t/(3f - t)$ and $2t/(-3f - t)$ respectively. Division of $b = 16b'$ by 2^6 gives the result. □

b_4: $n \equiv 3 \bmod 72$, $q \equiv 1 \bmod 18$. $(t, f) \equiv (2, 2) \bmod 3$ is the only option, so $b' = 16$ into Step 3 of Alg. C.2 gives $16^{(q-1)/6} \equiv 2t/(-3f - t) \bmod q$. Division of $b = 16b'$ by 2^6 finishes the proof. □

b_2: $n \equiv 243 \bmod 432$. $(t, f) \equiv (2, 0) \bmod 3$ means $b = 16$ from Step 1 of Alg. C.2. Division by $8 = \mu^3$ for $\mu = \sqrt{2}$ gives an isomorphic curve with $b = 2$. □

b_{-5}: $n \equiv 3 \bmod 360$, $q \equiv 1267 \bmod 1620$. $(t, f) \equiv (2, 1) \bmod 3$ is the only option, so Step 3 with $b' = -20$ yields $-20^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$. Division of $b = 16b'$ by 2^6 gives the result. □

b_3: $n \equiv 3 \bmod 24$, $q \equiv 1 \bmod 12$. Three cases arise: $(t, f) \equiv (2, 0) \bmod 3$ means $b = 16$. It isn't hard to show $3/16 = \mu^6$ for $\mu \in \mathbb{F}_q$ so that $b = 3$ gives an isomorphic curve. The other two cases are $(t, f) \equiv (2, 1) \bmod 3$ and $(t, f) \equiv (2, 2) \bmod 3$, both of which use $b' = 12$ in Step 3 of Alg. C.2 to give $12^{(q-1)/6} \bmod q$ as $2t/(3f - t)$ and $2t/(-3f - t)$ respectively. Division of $b = 16b'$ by 2^6 gives an isomorphic curve and finishes the proof. □

b_9: $n \equiv 3 \bmod 24$, $q \equiv 1 \bmod 6$. Two cases: $(t, f) \equiv (2, 0) \bmod 3$ means $b = 16$, for which it isn't hard to show $9/16 = \mu^3$ (and hence $\tilde{\mu}^6$), giving $b = 9$ as an isomorphic curve. □

b_5: $n \equiv 3 \bmod 24$, $q \equiv 1 \bmod 30$. Two cases: $(t, f) \equiv (2, 0) \bmod 3$ means $b = 16$. Again we use $5/16 = \mu^6$ for some μ to give the smaller constant. For the second case, $(t, f) \equiv (2, 1) \bmod 3$, so Step 3 of Alg. C.2 with $b' = 20$ gives $20^{(q-1)/6} \equiv 2t/(3f - t) \bmod q$, and division of $b = 16b'$ by 2^6 finishes the proof. □

b_2: $n \equiv 243 \bmod 432$. $(t, f) \equiv (2, 0) \bmod 3$ is the only case, which immediately gives $b = 16$ from Step 1 of Alg. C.2. $(\frac{2}{q})_2 = 1$ is easy (Prop. C.2), so $8 = \mu^6$ and $b = 2$ is a smaller constant. □

Appendix D

Some more generators

For the sake of protocols or implementations that may require them, this section lists extra generators that were found in the pairing groups $\mathbb{G}_1$ and $\mathbb{G}_2$ in each of the subfamilies. For the most part we stopped looking for any more once we had found 2 or 3 extra generators in any subfamily.

D.1 More compact generators for $k = 8$

Refer back to Table 6.2 - (i) : In $\mathbb{G}_2$, we also have $[h'](2/u, \sqrt{-4/u - 1})$. (ii) : In $\mathbb{G}_2$, $[h'](u - 3, \sqrt{(u-3)^3 + (u-3)u}$, $[h'](u + 2, \sqrt{(u+2)^3 + u(u+2)})$. (iii) : In $\mathbb{G}_1$, $(-1, \sqrt{1})$, $(-2, \sqrt{-4})$, $(2, 2)$. In $\mathbb{G}_2$, $(u + 2, \sqrt{(u+2)^3 - 2(u+2)/u})$ and $(u - 3, \sqrt{(u-3)^3 - 2(u-3)/u}$ also work. (iv) : In $\mathbb{G}_2$ is $[h'](-1, \sqrt{-1 - 2u})$. (v) : In $\mathbb{G}_2$ we also have $[h](-3, 2\sqrt{-6})$, $[h](-1, \sqrt{2})$ and $[h](3, \sqrt{30})$; $\mathbb{G}_2$ also has $[h'](-1, -1 + 3/u)$. (vi) : $\mathbb{G}_1$ also has $[h](-4, 6\sqrt{-2})$. (vii) : Again, $\mathbb{G}_1$ also has $[h](-4, 6\sqrt{-2})$. $(viii)$: $\mathbb{G}_1$ also has $[h](-2, 2\sqrt{-3})$. (ix) : Again, $\mathbb{G}_1$ has $[h](-2, 2\sqrt{-3})$ too. $\mathbb{G}_2$ also has $[h']\left(-5, \sqrt{-125 - 25/u}\right)$.

D.2 More compact generators for $k = 12$

Refer back to Table 6.4 - (i) : In $\mathbb{G}_1$, we also have $[h']\left(-5, \sqrt{-128}\right)$, $[h']\left(3, \sqrt{24}\right)$ and $[h']\left(9, \sqrt{726}\right)$.

D.3 More compact generators for $k = 18$

Refer back to Table 6.8 - for all cases here, the extra generators are in $\mathbb{G}_1$: (i) $[h](-3, \sqrt{-25})$, $[h](1, \sqrt{3})$; (ii) $[h](-1, \sqrt{3})$; (iii) $[h](-2, 4\sqrt{-3})$, $[h](1, \sqrt{-3})$, $[h](5, 11)$; (iv) $[h](-3, 2\sqrt{-6})$, $[h](-1, \sqrt{2})$; (v) $[h](-2, 2\sqrt{-3})$, $[h](1, \sqrt{-3})$; (vi) $[h](-5, 2\sqrt{-30})$, $[h](-2, \sqrt{-3})$; (vii) $[h](-1, 2\sqrt{-2})$.

D.4 More compact generators for $k = 27$

Refer back to Table 6.10 - all the extra generators are in $\mathbb{G}_1$: (i) : $[h](-5, 8\sqrt{-2})$, $[h](3, 2\sqrt{6})$ $[h](9, 11\sqrt{6})$; (ii): $[h](7, 4\sqrt{21})$; (iii) : $[h](3, 6)$, $[h](6, 3\sqrt{5})$.

D.5 More compact generators for $k = 32$

Refer back to Table 6.12 - all the extra generators are in $\mathbb{G}_1$: (i): $[h](-5, 8\sqrt{-2})$, $[h](3, 2\sqrt{6})$, $[h](9, 11\sqrt{6})$; (ii): $[h](-4, 6\sqrt{2})$; (iii): $[h](-3, 6\sqrt{-1})$, $[h](-1, 2\sqrt{-1})$, $[h](3, 6)$.

D.6 More compact generators for $k = 36$

Refer back to Table 6.14 - all the extra generators are in $\mathbb{G}_1$: (i): $[h](-2, 2\sqrt{-3})$, $[h](1, \sqrt{-3})$, $[h](5, 11)$; (ii): $[h](-2, \sqrt{-10})$, $[h](-1, \sqrt{-3})$; (iii): $[h](-2, \sqrt{-5})$; (iv): $[h](3, \sqrt{30})$; (v): $[h](5, 2\sqrt{30})$; (vi): $[h](-2, 2\sqrt{-3})$, $[h](-1, \sqrt{-5})$, $[h](1, \sqrt{-3})$, $[h](4, 2\sqrt{15})$.

D.7 More compact generators for $k = 48$

Refer back to Table 6.16 - (i): Both in $\mathbb{G}_2$ are $[h'](-1-2/w, \sqrt{(-1-2/w)^3 - 2})$, $[h'](\pm 5 - 2/w, \sqrt{(\pm 5 - 2/w)^3 - 2})$; (ii): In $\mathbb{G}_2$ is $[h'](1 - w, \sqrt{(1-w)^3 + 1})$; (iii): In $\mathbb{G}_2$ is $[h'](-2, \sqrt{-8 + 4w})$; (iv): All in $\mathbb{G}_2$ are $[h'](-1, \sqrt{-1 + 4w})$, $[h'](-3, \sqrt{-27 + 4w})$, $[h'](3, \sqrt{27 + 4w})$; (v): In $\mathbb{G}_2$ is $[h'](-1, \sqrt{1 - w})$; (vi): All in $\mathbb{G}_1$ are $[h](-5, 11\sqrt{1})$, $[h](-2, 2\sqrt{-1})$, $[h](2, 2\sqrt{3})$.

Bibliography

[AB10] M. Abdalla and P. S. L. M. Barreto, editors. *Progress in Cryptology - LATINCRYPT 2010, First International Conference on Cryptology and Information Security in Latin America, Puebla, Mexico, August 8-11, 2010, Proceedings*, volume 6212 of *Lecture Notes in Computer Science*. Springer, 2010.

[ACD+05] R. M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *The Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC, 2005.

[AFCK+12] D. F. Aranha, L. Fuentes-Castañeda, E. Knapp, A. J. Menezes, and F. Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. Cryptology ePrint Archive, Report 2012/232, 2012. `http://eprint.iacr.org/`.

[AKL+11] D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López. Faster explicit formulas for computing pairings over ordinary curves. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 48–68. Springer, 2011.

[AKM06] F. K. Abu Salem and K. Khuri-Makdisi. Fast Jacobian group operations for $C_{3,4}$ curves over a large finite field. *CoRR*, abs/math/0610121, 2006.

[AKMRH11] D. F. Aranha, E. Knapp, A. Menezes, and F. Rodríguez-Henríquez. Parallelizing the Weil and Tate pairings. In Chen [Che11], pages 275–295.

[ALNR10] C. Arène, T. Lange, M. Naehrig, and C. Ritzenthaler. Faster computation of the Tate pairing. *Journal of Number Theory*, 131(5):842–857, 2010.

[AM93] A.O.L. Atkin and F. Morain. Elliptic curves and primality proving. *Mathematics of computation*, 61:29–29, 1993.

[ANM09] M. Akane, Y. Nogami, and Y. Morikawa. Fast ate pairing computation of embedding degree 12 using subfield-twisted elliptic curve. *IEICE Transactions*, 92-A(2):508–516, 2009.

[ATW08] R. Avanzi, N. Thériault, and Z. Wang. Rethinking low genus hyperelliptic Jacobian arithmetic over binary fields: interplay of field arithmetic and explicit formulæ. *Journal of Mathematical Cryptology*, 2(3):227–255, 2008.

[BB11] D. Boneh and X. Boyen. Efficient selective identity-based encryption without random oracles. *J. Cryptology*, 24(4):659–693, 2011.

[BBC+09] J. Balakrishnan, J. Belding, S. Chisholm, K. Eisenträger, K.E. Stange, and E. Teske. Pairings on hyperelliptic curves. *WIN-Women in Numbers: Research Directions in Number Theory, Fields Institute Communications*, 60:87–120, 2009.

[BCP97] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[Ben10] N. Benger. *Cryptographic Pairings: Efficiency and DLP Security*. PhD thesis, Dublin City University, May 2010.

[Ber01] D.J. Bernstein. Multidigit multiplication for mathematicians. *Advances in Applied Mathematics*, 2001.

[Ber06] D. J. Bernstein. Elliptic vs. Hyperelliptic, part I. Talk at ECC, September 2006.

[BF03] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[BGDM+10] J. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya. High-speed software implementation of the optimal ate pairing over Barreto-Naehrig curves. In Joye et al. [JMO10], pages 21–39.

[BGN05] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.

[BGOS07] P. S. L. M. Barreto, S. D. Galbraith, C. O'Eigeartaigh, and M. Scott. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography*, 42(3):239–271, 2007.

[BJ03] O. Billet and M. Joye. The Jacobi model of an elliptic curve and side-channel analysis. In M. P. C. Fossorier, T. Hoholdt, and A. Poli, editors, *AAECC*, volume 2643 of *Lecture Notes in Computer Science*, pages 34–42. Springer, 2003.

[BK98] R. Balasubramanian and N. Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the Menezes - Okamoto - Vanstone algorithm. *J. Cryptology*, 11(2):141–145, 1998.

[BKLS02] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In Yung [Yun02], pages 354–368.

[BKMX06] I.F. Blake, V. Kumar Murty, and G. Xu. Refinements of Miller's algorithm for computing the Weil/Tate pairing. *Journal of Algorithms*, 58(2):134–149, 2006.

[BL07a] D. J. Bernstein and T. Lange. Explicit-formulas database. `http://www.hyperelliptic.org/EFD`, 2007.

[BL07b] D. J. Bernstein and T. Lange. Faster addition and doubling on elliptic curves. In K. Kurosawa, editor, *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.

[BLS02] P. S. L. M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In S. Cimato, C. Galdi,

and G. Persiano, editors, *SCN*, volume 2576 of *Lecture Notes in Computer Science*, pages 257–267. Springer, 2002.

[BLS03] P. S. L. M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In M. Matsui and R. J. Zuccherato, editors, *Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 2003.

[BLS04] P. S. L. M. Barreto, B. Lynn, and M. Scott. Efficient implementation of pairing-based cryptosystems. *J. Cryptology*, 17(4):321–334, 2004.

[BMW05] X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. In V. Atluri, C. Meadows, and A. Juels, editors, *ACM Conference on Computer and Communications Security*, pages 320–329. ACM, 2005.

[BN05] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In B. Preneel and S. E. Tavares, editors, *Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.

[BRS11] D. Boneh, K. Rubin, and A. Silverberg. Finding composite order ordinary elliptic curves using the Cocks-Pinch method. *Journal of Number Theory*, 131(5):832–841, 2011.

[BS10] N. Benger and M. Scott. Constructing tower extensions of finite fields for implementation of pairing-based cryptography. In Hasan and Helleseth [HH10], pages 180–195.

[BW05] F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. *Des. Codes Cryptography*, 37(1):133–141, 2005.

[CA66] S.A. Cook and S.O. Aanderaa. *On the minimum computation time of functions.* PhD thesis, Harvard., 1966.

[Can87] D. G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Mathematics of computation*, 48(177):95–101, January 1987.

[CBNW10a] C. Costello, C. Boyd, J. M. González Nieto, and K. Koon-Ho Wong. Avoiding full extension field arithmetic in pairing computations. In D. J. Bernstein and T. Lange, editors, *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 203–224. Springer, 2010.

[CBNW10b] C. Costello, C. Boyd, J. M. González Nieto, and K. Koon-Ho Wong. Delaying mismatched field multiplications in pairing computations. In Hasan and Helleseth [HH10], pages 196–214.

[CCS07] L. Chen, Z. Cheng, and N. P. Smart. Identity-based key agreement protocols from pairings. *Int. J. Inf. Sec.*, 6(4):213–241, 2007.

[CHB+09] C. Costello, H. Hisil, C. Boyd, J. M. González Nieto, and K. Koon-Ho Wong. Faster pairings on special Weierstrass curves. In Shacham and Waters [SW09], pages 89–101.

[Che11] L. Chen, editor. *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, volume 7089 of *Lecture Notes in Computer Science*. Springer, 2011.

[CJLM06] M. Ciet, M. Joye, K. Lauter, and P. L. Montgomery. Trading inversions for multiplications in elliptic curve cryptography. *Des. Codes Cryptography*, 39(2):189–206, 2006.

[CK03] L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairings. In *CSFW*, pages 219–233. IEEE Computer Society, 2003.

[CL11] C. Costello and K. Lauter. Group law computations on Jacobians of hyperelliptic curves. In Miri and Vaudenay [MV12], pages 92–117.

[CLN10] C. Costello, T. Lange, and M. Naehrig. Faster pairing computations on curves with high-degree twists. In Nguyen and Pointcheval [NP10], pages 224–242.

[CLN11] C. Costello, K. Lauter, and M. Naehrig. Attractive subfamilies of BLS curves for implementing high-security pairings. In D. J. Bernstein and S. Chatterjee, editors, *INDOCRYPT*, volume 7107

of *Lecture Notes in Computer Science*, pages 320–342. Springer, 2011.

[CM09] S. Chatterjee and A. J. Menezes. On cryptographic protocols employing asymmetric pairings - the role of psi revisited. *IACR Cryptology ePrint Archive*, 2009:480, 2009.

[Coh96] H. Cohen. *A course in computational algebraic number theory*, volume 138. Springer-Verlag, 3rd printing, 1996.

[Cos12] C. Costello. Particularly friendly members of family trees. Cryptology ePrint Archive, Report 2012/072, 2012. `http://eprint.iacr.org/`.

[CP01] C. Cocks and R.G.E. Pinch. Id-based cryptosystems based on the Weil pairing. Unpublished manuscript, 2001.

[CS10] C. Costello and D. Stebila. Fixed argument pairings. In Abdalla and Barreto [AB10], pages 92–108.

[CSB04] S. Chatterjee, P. Sarkar, and R. Barua. Efficient computation of Tate pairing in projective coordinate over general characteristic fields. In C. Park and S. Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 168–181. Springer, 2004.

[CV11] W. Castryck and F. Vercauteren. Toric forms of elliptic curves and their arithmetic. *J. Symb. Comput.*, 46(8):943–966, 2011.

[DEM05] R. Dupont, A. Enge, and F. Morain. Building curves with arbitrary small MOV degree over finite prime fields. *J. Cryptology*, 18(2):79–89, 2005.

[Deu41] M. Deuring. Die typen der multiplikatorenringe elliptischer funktionenkörper. *Abh. Math. Sem. Hansischen Univ.*, 14:197–242, 1941.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[Die06] C. Diem. An index calculus algorithm for plane curves of small degree. In Hess et al. [HPP06], pages 543–557.

[Die12] C. Diem. What on earth is "index calculus"? The ECC blog: http://ellipticnews.wordpress.com/2012/05/07/246/, May 2012.

[DIK06] C. Doche, T. Icart, and D. R. Kohel. Efficient scalar multiplication by isogeny decompositions. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 191–206. Springer, 2006.

[DKS09] L. J. Dominguez Perez, E. J. Kachisa, and M. Scott. Implementing cryptographic pairings: a magma tutorial. Cryptology ePrint Archive, Report 2009/072, 2009. http://eprint.iacr.org/.

[DL03] I. M. Duursma and H. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In C. Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 111–123. Springer, 2003.

[DOSD06] A. J. Devegili, C. O'Eigeartaigh, M. Scott, and R. Dahab. Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Report 2006/471, 2006. http://eprint.iacr.org/.

[DS08] M. P. L. Das and P. Sarkar. Pairing computation on twisted Edwards form elliptic curves. In Galbraith and Paterson [GP08], pages 192–210.

[DS10] L. J. Dominguez Perez and M. Scott. Private communication, November 2010.

[DSD07] A. J. Devegili, M. Scott, and R. Dahab. Implementing cryptographic pairings over Barreto-Naehrig curves. In Takagi et al. [TOOO07], pages 197–207.

[Edw07] H.M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44(3):393–422, 2007.

[EJSS10] S. Erickson, M. J. Jacobson, N. Shang, and S. Shen A. Stein. Efficient formulas for real hyperelliptic curves of genus 2 in affine representation. In C. Carlet and B. Sunar, editors, *Arithmetic of*

finite fields, volume 4547 of *Lecture Notes in Computer Science*, pages 202–218. Springer Berlin / Heidelberg, 2010.

[ELM03] K. Eisenträger, K. Lauter, and P. L. Montgomery. Fast elliptic curve arithmetic and improved Weil pairing evaluation. In Joye [Joy03], pages 343–354.

[ELM04] K. Eisenträger, K. Lauter, and P. L. Montgomery. Improved Weil and Tate pairings for elliptic and hyperelliptic curves. In D. A. Buell, editor, *ANTS*, volume 3076 of *Lecture Notes in Computer Science*, pages 169–183. Springer, 2004.

[FCKRH11] L. Fuentes-Castañeda, E. Knapp, and F. Rodríguez-Henríquez. Faster hashing to G_2. In Miri and Vaudenay [MV12], pages 412–430.

[FGJ08] X. Fan, G. Gong, and D. Jao. Efficient pairing computation on genus 2 curves in projective coordinates. In R. M. Avanzi, L. Keliher, and F. Sica, editors, *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2008.

[FO04] S. Flon and R. Oyono. Fast arithmetic on Jacobians of Picard curves. In F. Bao, R. H. Deng, and J. Zhou, editors, *Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 55–68. Springer, 2004.

[FOR08] S. Flon, R. Oyono, , and C. Ritzenthaler. Fast addition on non-hyperelliptic genus 3 curves. *Algebraic geometry and its applications*, 5(3):227–256, 2008.

[FR94] G. Frey and H.G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of computation*, 62(206):865–874, 1994.

[Fre06] D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In Hess et al. [HPP06], pages 452–465.

[Fre10] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Gilbert [Gil10], pages 44–61.

[Fri05] S. Friedl. An elementary proof of the group law for elliptic curves. Personal webpage: `http://math.rice.edu/~friedl/papers/AAELLIPTIC.PDF`, August 2005.

[FST10] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2):224–280, 2010.

[Ful08] W. Fulton. *Algebraic curves: an introduction to algebraic geometry (3rd edition).* `http://www.math.lsa.umich.edu/~wfulton/CurveBook.pdf`, 2008.

[Gal01] S. D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2001.

[Gal05] S. D. Galbraith. *Pairings*, volume 317 of *London Mathematical Society Lecture Notes*, chapter IX, pages 183–213. Cambridge University Press, 2005.

[Gal09] S. D. Galbraith. Twists of Edwards curves. unpublished manuscript, 2009.

[Gal12] S. D. Galbraith. *Mathematics of Public Key Cryptography.* Cambridge University Press, March 2012.

[Gau00] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In B. Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2000.

[Gau05] P. Gaudry. *Hyperelliptic curves and the HCDLP*, volume 317 of *London Mathematical Society Lecture Notes*, chapter VII, pages 133–150. Cambridge University Press, 2005.

[Gau07] P. Gaudry. Fast genus 2 arithmetic based on Theta functions. *Journal of Mathematical Cryptology*, 1(3):243–265, 2007.

[GH00] P. Gaudry and R. Harley. Counting points on hyperelliptic curves over finite fields. In W. Bosma, editor, *ANTS*, volume 1838 of *Lecture Notes in Computer Science*, pages 313–332. Springer, 2000.

[GHM08] S. D. Galbraith, M. Harrison, and D. J. Mireles Morales. Efficient hyperelliptic arithmetic using balanced representation for divisors. In A. J. van der Poorten and A. Stein, editors, *ANTS*, volume 5011 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2008.

[GHO+07] R. Granger, F. Hess, R. Oyono, N. Thériault, and F. Vercauteren. Ate pairing on hyperelliptic curves. In M. Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 430–447. Springer, 2007.

[GHS02] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In C. Fieker and D. R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 2002.

[GHV07] S. D. Galbraith, F. Hess, and F. Vercauteren. Hyperelliptic pairings. In Takagi et al. [TOOO07], pages 108–131.

[Gil10] H. Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[GKP04] C. Gurot, K. Kaveh, and V. M. Patankar. Explicit algorithm for the arithmetic on the hyperelliptic Jacobians of genus 3. *Journal of the Ramanujan Mathematical Society*, 19:75–115, 2004.

[GL09] S. D. Galbraith and X. Lin. Computing pairings using x-coordinates only. *Designs, Codes and Cryptography*, 50(3):305–324, 2009.

[GLS11] S. D. Galbraith, X. Lin, and M. Scott. Endomorphisms for faster elliptic curve cryptography on a large class of curves. *J. Cryptology*, 24(3):446–469, 2011.

[GLV01] R. P. Gallant, R. J. Lambert, and S. A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In J. Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2001.

[GMA+05] M. Gonda, K. Matsuo, K. Aoki, J. Chao, and S. Tsujii. Improvements of addition algorithm on genus 3 hyperelliptic curves and their implementation. *IEICE Transactions*, 88-A(1):89–96, 2005.

[GMV07] S. D. Galbraith, J. F. McKee, and P. C. Valença. Ordinary abelian varieties having small embedding degree. *Finite Fields and Their Applications*, 13(4):800–814, 2007.

[GP08] S. D. Galbraith and K. G. Paterson, editors. *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, volume 5209 of *Lecture Notes in Computer Science*. Springer, 2008.

[GPS06] R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing-based cryptography. *LMS Journal of Computation and Mathematics*, 9(1):64–85, 2006.

[GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[GPSW06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[GS10] R. Granger and M. Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. In Nguyen and Pointcheval [NP10], pages 209–223.

[GTTD07] P. Gaudry, E. Thomé, N Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comput.*, 76(257):475–492, 2007.

[Har] R. Harley. Fast arithmetic on genus 2 curves. See http://cristal.inria.fr/∼harley/hyper for `C` source code and further explanations.

[Har77] R. Hartshorne. *Algebraic Geometry*, volume 52 of *Graduate texts in mathematics*. Springer-Verlag, 1977.

[Hes02] F. Hess. Computing Riemann-Roch spaces in algebraic function fields and related topics. *J. Symb. Comput.*, 33(4):425–445, 2002.

[Hes08] F. Hess. Pairing lattices. In Galbraith and Paterson [GP08], pages 18–38.

[HH10] M. A. Hasan and T. Helleseth, editors. *Arithmetic of Finite Fields, Third International Workshop, WAIFI 2010, Istanbul, Turkey, June 27-30, 2010. Proceedings*, volume 6087 of *Lecture Notes in Computer Science*. Springer, 2010.

[HI94] M. A. Huang and D. Ierardi. Efficient algorithms for the Riemann-Roch problem and for addition in the jacobian of a curve. *J. Symb. Comput.*, 18(6):519–539, 1994.

[His10] H. Hisil. *Elliptic curves, group law, and efficient computation*. PhD thesis, Queensland University of Technology, 2010.

[HLX12] Z. Hu, P. Longa, and M. Xu. Implementing the 4-dimensional GLV method on GLS elliptic curves with j-invariant 0. *Des. Codes Cryptography*, 63(3):331–343, 2012.

[HMS08] D. Hankerson, A. J. Menezes, and M. Scott. Software implementation of pairings. In M. Joye and G. Neven, editors, *Identity-Based Cryptography*, pages 188–206. IOS Press, 2008.

[HPP06] F. Hess, S. Pauli, and M. E. Pohst, editors. *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings*, volume 4076 of *Lecture Notes in Computer Science*. Springer, 2006.

[HSV06] F. Hess, N. P. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[HWCD08] H. Hisil, K. Koon-Ho Wong, G. Carter, and E. Dawson. Twisted Edwards curves revisited. In J. Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2008.

[HWCD09] H. Hisil, K. Koon-Ho Wong, G. Carter, and E. Dawson. Jacobi quartic curves revisited. In C. Boyd and J. M. González Nieto, editors, *ACISP*, volume 5594 of *Lecture Notes in Computer Science*, pages 452–468. Springer, 2009.

[IJ08] S. Ionica and A. Joux. Another approach to pairing computation in Edwards coordinates. In D. R. Chowdhury, V. Rijmen, and A. Das, editors, *INDOCRYPT*, volume 5365 of *Lecture Notes in Computer Science*, pages 400–413. Springer, 2008.

[IR90] K. Ireland and M. Rosen. *A Classical Introduction to Modern Number Theory*, volume 84 of *Graduate texts in mathematics*. Springer-Verlag, 1990.

[IT02] T. Izu and T. Takagi. Efficient computations of the Tate pairing for the large MOV degrees. In P. J. Lee and C. H. Lim, editors, *ICISC*, volume 2587 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2002.

[JMO10] M. Joye, A. Miyaji, and A. Otsuka, editors. *Pairing-Based Cryptography - Pairing 2010 - 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings*, volume 6487 of *Lecture Notes in Computer Science*. Springer, 2010.

[Jou04] A. Joux. A one round protocol for tripartite Diffie-Hellman. *J. Cryptology*, 17(4):263–276, 2004.

[Joy03] M. Joye, editor. *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings*, volume 2612 of *Lecture Notes in Computer Science*. Springer, 2003.

[JQ01] M. Joye and J.J. Quisquater. Hessian elliptic curves and side-channel attacks. In *Cryptographic Hardware and Embedded Systems—CHES 2001*, pages 402–410. Springer, 2001.

[JTV10] M. Joye, M. Tibouchi, and D. Vergnaud. Huff's model for elliptic curves. In G. Hanrot, F. Morain, and E. Thomé, editors, *ANTS*, volume 6197 of *Lecture Notes in Computer Science*, pages 234–250. Springer, 2010.

[Kar10] K. Karabina. Squaring in cyclotomic subgroups. *IACR Cryptology ePrint Archive*, 2010:542, 2010.

[KKAT04] M. Katagi, I. Kitamura, T. Akishita, and T. Takagi. Novel efficient implementations of hyperelliptic curve cryptosystems using degenerate divisors. In C. H. Lim and M. Yung, editors, *WISA*, volume 3325 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2004.

[KM04] K. Khuri-Makdisi. Linear algebra algorithms for divisors on an algebraic curve. *Math. Comput.*, 73(245):333–357, 2004.

[KM05] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In N. P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 13–36. Springer, 2005.

[KM07] K. Khuri-Makdisi. Asymptotically fast group operations on Jacobians of general curves. *Math. Comput.*, 76(260):2213–2239, 2007.

[KO63] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. In *Soviet physics doklady*, volume 7, page 595, 1963.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[Kob89] N. Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1(3):139–150, 1989.

[Koh11] D. Kohel. Addition law structure of elliptic curves. *Journal of Number Theory*, 2011.

[KSS08] E. J. Kachisa, E. F. Schaefer, and M. Scott. Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In Galbraith and Paterson [GP08], pages 126–135.

[Lan72] S. Lang. *Introduction to algebraic geometry*. Addison-Wesley, 1972.

[Lan01] T. Lange. *Efficient arithmetic on hyperelliptic curves*. PhD thesis, Universität-Gesamthochschule Essen, 2001.

[Lan02a] T. Lange. Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae. Cryptology ePrint Archive, Report 2002/121, 2002. `http://eprint.iacr.org/`.

[Lan02b] T. Lange. Inversion-free arithmetic on genus 2 hyperelliptic curves. Cryptology ePrint Archive, Report 2002/147, 2002. `http://eprint.iacr.org/`.

[Lan02c] T. Lange. Weighted coordinates on genus 2 hyperelliptic curves. Cryptology ePrint Archive, Report 2002/153, 2002. `http://eprint.iacr.org/`.

[Lan05] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Appl. Algebra Eng. Commun. Comput.*, 15(5):295–328, 2005.

[Lan06] T. Lange. Elliptic vs. Hyperelliptic, part II. Talk at ECC, September 2006.

[Lau03] K. Lauter. The equivalence of the geometric and algebraic group laws for Jacobians of genus 2 curves. In *Topics in algebraic and noncommutative geometry: proceedings in memory of Ruth Michler, July 20-22, 2001, Luminy, France [and] October 25-28, 2001, Annapolis, Maryland*, volume 324, page 165. Amer Mathematical Society, 2003.

[LD98] J. López and R. Dahab. Improved algorithms for elliptic curve arithmetic in GF(2^n). In S. E. Tavares and H. Meijer, editors, *Selected Areas in Cryptography*, volume 1556 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 1998.

[Leh58] E. Lehmer. Criteria for cubic and quartic residuacity. *Mathematika*, 5(20-29), 1958.

[Lei05] F. Leitenberger. About the group law for the Jacobi variety of a hyperelliptic curve. *Contributions to Algebra and Geometry*, 46(1):125–130, 2005.

[Lem00] F. Lemmermeyer. *Reciprocity laws: from Euler to Eisenstein*. Springer Verlag, 2000.

[Lew12] A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 318–335. Springer, 2012.

[Lic69] S. Lichtenbaum. Duality theorems for curves over P-adic fields. *Inventiones mathematicae*, 7(2):120–136, 1969.

[LLP09] E. Lee, H.-S. Lee, and C.-M. Park. Efficient and generalized pairing computation on abelian varieties. *IEEE Transactions on Information Theory*, 55(4):1793–1803, 2009.

[LMN10] K. Lauter, P. L. Montgomery, and M. Naehrig. An analysis of affine coordinates for pairing computation. In Joye et al. [JMO10], pages 1–20.

[LOS+10] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Gilbert [Gil10], pages 62–91.

[Lyn07] B. Lynn. *On the Efficient Implementation of Pairing-Based Cryptosystems*. PhD thesis, Stanford University, June 2007.

[LZZW08] X. Lin, C. Zhao, F. Zhang, and Y. Wang. Computing the ate pairing on elliptic curves with embedding degree $k = 9$. *IEICE Transactions*, 91-A(9):2387–2393, 2008.

[MB05] N. McCullagh and P. S. L. M. Barreto. A new two-party identity-based authenticated key agreement. In Menezes [Men05], pages 262–274.

[MCT01] K. Matsuo, J. Chao, and S. Tsujii. Fast genus two hyperelliptic curve cryptosystems. Technical Report 214, IEIC, 2001.

[MDM+02] Y. Miyamoto, H. Doi, K. Matsuo, J. Chao, and S. Tsujii. A fast addition algorithm of genus two hyperelliptic curve. In *Symposium on Cryptography and Information Security - SCICS*, In Japanese, 2002.

[Men93] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, 1993.

[Men05] A. J. Menezes, editor. *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*. Springer, 2005.

[Men09] A. J. Menezes. Asymmetric Pairings. Talk at ECC 2009, University of Calgary, Canada., August 2009.

[MGI09] N. El Mrabet, N. Guillermin, and S. Ionica. A study of pairing computation for elliptic curves with embedding degree 15. Cryptology ePrint Archive, Report 2009/370, 2009. `http://eprint.iacr.org/`.

[Mil85] V. S. Miller. Use of elliptic curves in cryptography. In H. C. Williams, editor, *CRYPTO*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.

[Mil04] V. S. Miller. The Weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.

[Min10] H. Minkowski. *Geometrie der zahlen*, volume 1896. Teubner, 1910.

[MKHO07] S. Matsuda, N. Kanayama, F. Hess, and E. Okamoto. Optimised versions of the ate and twisted ate pairings. In S. D. Galbraith, editor, *IMA Int. Conf.*, volume 4887 of *Lecture Notes in Computer Science*, pages 302–312. Springer, 2007.

[MNT01] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 2001.

[Mon87] P.L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264, 1987.

[MOV93] A. J. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[MS07] R. Murty and I. Shparlinski. Group structure of elliptic curves over finite fields and applications. *Topics in Geometry, Coding Theory and Cryptography*, pages 167–194, 2007.

[Mum84] D. Mumford. Tata lectures on theta II. In *Progress in Mathematics*, volume 43. Birkhiauser Boston Inc., Boston, MA, 1984.

[MV12] A. Miri and S. Vaudenay, editors. *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*. Springer, 2012.

[Nae09] M. Naehrig. *Constructive and computational aspects of cryptographic pairings*. PhD thesis, Eindhoven University of Technology, May 2009.

[NAS+08] Y. Nogami, M. Akane, Y. Sakemi, H. Katou, and Y. Morikawa. Integer variable chi-based ate pairing. In Galbraith and Paterson [GP08], pages 178–191.

[NBS08] M. Naehrig, P. S. L. M. Barreto, and P. Schwabe. On compressible pairings and their computation. In S. Vaudenay, editor, *AFRICACRYPT*, volume 5023 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2008.

[NIS99] NIST. Recommended elliptic curves for Federal Government Use. Technical report, National Institute of Standards and Technology, July 1999.

[NNS10] M. Naehrig, R. Niederhagen, and P. Schwabe. New software speed records for cryptographic pairings. In Abdalla and Barreto [AB10], pages 109–123.

[NP10] P. Q. Nguyen and D. Pointcheval, editors. *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, volume 6056 of *Lecture Notes in Computer Science*. Springer, 2010.

[Oka12] K. Okano. On conditions for ρ-value is 1 or not of complete family of pairing-friendly elliptic curves. `http://arxiv.org/abs/1205.1646`, May 2012.

[Pat05] K. G. Paterson. *Cryptography from Pairings*, volume 317 of *London Mathematical Society Lecture Notes*, chapter X, pages 215–251. Cambridge University Press, 2005.

[PJNB11] G. C. C. F. Pereira, M. A. Simplício Jr., M. Naehrig, and P. S. L. M. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8):1319–1326, 2011.

[Pol78] J.M. Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of computation*, 32(143):918–924, 1978.

[RS02] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In Yung [Yun02], pages 336–353.

[RS10] K. Rubin and A. Silverberg. Choosing the correct elliptic curve in the CM method. *Math. Comput.*, 79(269):545–561, 2010.

[SB04] M. Scott and P. S. L. M. Barreto. Compressed pairings. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2004.

[SB06] M. Scott and P. S. L. M. Barreto. Generating more MNT elliptic curves. *Des. Codes Cryptography*, 38(2):209–217, 2006.

[SBC+09a] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and Ezekiel J. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In Shacham and Waters [SW09], pages 78–88.

[SBC+09b] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and Ezekiel J. Kachisa. Fast hashing to G_2 on pairing-friendly curves. In Shacham and Waters [SW09], pages 102–113.

[SCA06] M. Scott, N. Costigan, and W. Abdulwahab. Implementing cryptographic pairings on smartcards. In L. Goubin and M. Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 134–147. Springer, 2006.

[Sch85] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp*, 44(170):483–494, 1985.

[Sco04] M. Scott. Understanding the Tate pairing. Personal webpage: `http://www.computing.dcu.ie/~mike/tate.html`, 2004.

[Sco05a] M. Scott. Computing the Tate pairing. In Menezes [Men05], pages 293–304.

[Sco05b] M. Scott. Faster pairings using an elliptic curve with an efficient endomorphism. In S. Maitra, C. E. V. Madhavan, and R. Venkatesan, editors, *INDOCRYPT*, volume 3797 of *Lecture Notes in Computer Science*, pages 258–269. Springer, 2005.

[Sco07a] M. Scott. An introduction to pairings. Talk at ICE-EM RNSA 2007 Cryptography Workshop, Queensland University of Technology, Australia, June 2007.

[Sco07b] M. Scott. Efficient implementation of cryptographic pairings. Talk at ICE-EM RNSA 2007 Cryptography Workshop, Queensland University of Technology, Australia, June 2007.

[Sco07c] M. Scott. Implementing cryptographic pairings. In Tsuyoshi Takagi, Tatsuaki Okamoto, and Eiji Okamoto, editors, *Pairing-Based Cryptography – Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 177–196. Springer, 2007.

[Sco09] M. Scott. A note on twists for pairing friendly curves. Personal webpage: `ftp://ftp.computing.dcu.ie/pub/resources/crypto/twists.pdf`, February 2009.

[Sco11] M. Scott. On the efficient implementation of pairing-based protocols. In Chen [Che11], pages 296–308.

[Sha05] H. Shacham. *New Paradigms in Signature Schemes*. PhD thesis, Stanford University, December 2005.

[Sil09] J. H. Silverman. *The Arithmetic of Elliptic Curves (2nd Edition)*. Number 106 in Graduate texts in mathematics. Springer-Verlag, 2009.

[Sil10] J. H. Silverman. A survey of local and global pairings on elliptic curves and abelian varieties. In Joye et al. [JMO10], pages 377–396.

[Sma01] N. P. Smart. The Hessian form of an elliptic curve. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 118–125. Springer, 2001.

[Sma02] N. P. Smart. Identity-based authenticated key agreement protocol based on Weil pairing. *Electronics Letters*, 38(13):630–632, 2002.

[Sma10] N. P. Smart. ECRYPT II yearly report on algorithms and keysizes (2009-2010). Technical report, ECRYPT II – European Network of Excellence in Cryptology, EU FP7, ICT-2007-216676, 2010. Published as deliverable D.SPA.13, `http://www.ecrypt.eu.org/documents/D.SPA.13.pdf`.

[SMCT02] H. Sugizaki, K. Matsuo, J. Chao, and S. Tsujii. An extension of Harley addition algorithm for hyperelliptic curves over finite fields of characteristic two. Technical Report ISEC2002-9(2002-5), IEICE, 2002.

[Smi09] B. Smith. Isogenies and the discrete logarithm problem in Jacobians of genus 3 hyperelliptic curves' . *J. Cryptology*, 22(4):505–529, 2009.

[SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan*, pages 135–148, 2000.

[Sta07] K. E. Stange. The Tate pairing via elliptic nets. In Takagi et al. [TOOO07], pages 329–348.

[Sut12] A. V. Sutherland. Accelerating the CM method. *LMS Journal of Computation and Mathematics*, 15:172–204, 2012.

[SV07] N. P. Smart and F. Vercauteren. On computable isomorphisms in efficient asymmetric pairing-based systems. *Discrete Applied Mathematics*, 155(4):538–547, 2007.

[SW09] H. Shacham and B. Waters, editors. *Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings*, volume 5671 of *Lecture Notes in Computer Science*. Springer, 2009.

[Tak02] M Takahashi. Improving Harley algorithms for Jacobians of genus 2 hyperelliptic curves. In *Symposium on Cryptography and Information Security - SCICS*, In Japanese., 2002.

[Too63] A.L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, volume 3, pages 714–716, 1963.

[TOOO07] T. Takagi, T. Okamoto, E. Okamoto, and T. Okamoto, editors. *Pairing-Based Cryptography - Pairing 2007, First International Conference, Tokyo, Japan, July 2-4, 2007, Proceedings*, volume 4575 of *Lecture Notes in Computer Science*. Springer, 2007.

[Ver01] E. R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In B. Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2001.

[Ver06a] F. Vercauteren. Mathematics of Pairings: Part II. Talk at Pairing-Based Cryptography Workshop, 2006.

[Ver06b] F. Vercauteren. Mathematics of Pairings: Part I. Talk at Pairing-Based Cryptography Workshop, 2006.

[Ver10] F. Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.

[Wat05] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[WK07] T. Wollinger and V. Kovtun. Fast explicit formulae for genus 2 hyperelliptic curves using projective coordinates. In *ITNG*, pages 893–897. IEEE Computer Society, 2007.

[Wol04] T. Wollinger. *Software and hardware implementation of hyperelliptic curve cryptosystems*. PhD thesis, Ruhr-University of Bochum, 2004.

[WP06] A. Weimerskirch and C. Paar. Generalizations of the Karatsuba algorithm for efficient implementations. Cryptology ePrint Archive, Report 2006/224, 2006. `http://eprint.iacr.org/`.

[WPP05] T. J. Wollinger, J. Pelzl, and C. Paar. Cantor versus Harley: Optimization and analysis of explicit formulae for hyperelliptic curve cryptosystems. *IEEE Trans. Computers*, 54(7):861–872, 2005.

[WS07] C. Whelan and M. Scott. The importance of the final exponentiation in pairings when considering fault attacks. In Takagi et al. [TOOO07], pages 225–246.

[Yun02] M. Yung, editor. *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*. Springer, 2002.

[ZZH08a] C. Zhao, F. Zhang, and J. Huang. All pairings are in a group. Cryptology ePrint Archive, Report 2008/085, 2008. `http://eprint.iacr.org/`.

[ZZH08b] C. Zhao, F. Zhang, and J. Huang. A note on the ate pairing. *Int. J. Inf. Sec.*, 7(6):379–382, 2008.

Index