Security of Genus 3 Curves in Cryptography

by

Kim Henry Martin Laine

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Mathematics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kenneth A. Ribet, Chair
Professor Xinyi Yuan
Professor David Wagner

Summer 2015

Security of Genus 3 Curves in Cryptography

Abstract

Security of Genus 3 Curves in Cryptography

by

Kim Henry Martin Laine

Doctor of Philosophy in Mathematics

University of California, Berkeley

Professor Kenneth A. Ribet, Chair

The Discrete Logarithm Problem (DLP) in the abelian group $E(\mathbb{F}_p)$ of $\mathbb{F}_p$-valued points on an elliptic curve $E/\mathbb{F}_p$ has been successfully used as a building block for a wide variety of both public key encryption schemes and digital signature schemes. Since the size of $E(\mathbb{F}_p)$ is roughly p, to have a security level of at least 80 bits against generic collision attacks, such as Pollard rho, one must take p to be at least 160 bits.

Neil Koblitz suggested using instead the group of $\mathbb{F}_p$-valued points on the Jacobian Jac_C of a genus g hyperelliptic curve $C/\mathbb{F}_p$. The upshot is that the size of $\mathrm{Jac}_C(\mathbb{F}_p)$ is roughly p^g, so to obtain a security level of 80 bits against Pollard rho one needs to take p to be at least $160/g$ bits.

When $g = 2$ everything works out well. As in the case of elliptic curves, the most efficient known attack against the DLP is Pollard rho, which runs in time $\widetilde{O}(p)$. There are very efficient ways of doing arithmetic on the Jacobian of a genus 2 curve (or rather on the Kummer surface), and with the increase in efficiency coming from using smaller fields (p at least 80 bits), such genus 2 cryptosystems are fast enough to be competitive with elliptic curve cryptosystems.

When $g > 3$ there is an efficient index calculus algorithm for breaking the DLP, which is why such curves are not considered to be useful for cryptographic purposes. Moreover, there is no efficient enough way of doing arithmetic on such high-dimensional Jacobians.

When $g = 3$ the situation is much more complicated. There is an index calculus algorithm which breaks the DLP on the Jacobian of a hyperelliptic genus 3 curve in time $\widetilde{O}(p^{4/3})$. Note for comparison that Pollard rho runs in time $\widetilde{O}(p^{3/2})$. For practical field sizes (p at least 60 bits) the difference between these complexities is quite significant. Nevertheless, this index calculus algorithm might not be fast enough to be a significant threat and the practicality of the attack is highly debatable. But genus 3 hyperelliptic curves suffer also from another much worse security problem. Namely, it might be possible to compute an isogeny from the hyperelliptic Jacobian to another isogenous abelian variety, which has a good chance of being the Jacobian of a non-hyperelliptic genus 3 curve over the same base field. If such an isogeny can be computed explicitly enough, the DLP can be mapped to the Jacobian of the

non-hyperelliptic curve and solved using an even faster index calculus algorithm by Claus Diem, with complexity $\widetilde{O}(p)$. This sounds bad, but again the practicality of Diem's attack is debatable. The main problem turns out to be that both the hyperelliptic $\widetilde{O}(p^{4/3})$ and the non-hyperelliptic $\widetilde{O}(p)$ algorithms require vast amounts of memory. Therefore the security of genus 3 hyperelliptic curves depends strongly on the practical performance of Diem's index calculus and the feasibility of computing isogenies explicity.

We develop and study, both in theory and in practice, a new index calculus algorithm for attacking non-hyperelliptic genus 3 curves. Our attack is related to Diem's index calculus, but improves it in several aspects. We obtain detailed space and time complexity estimates for our algorithm, making all hidden factors in the $\widetilde{O}$-notation explicit. We discuss time-memory trade-offs and practical ways of dealing with the massive memory requirements. Finally we discuss ways of generating hyperelliptic genus 3 curves in such a way that performing isogeny attacks becomes particularly difficult. This is done by generating input data for the genus 3 CM method of Weng in a very careful way, allowing us to precisely control which isogenies the attacker can use to reach non-hyperelliptic Jacobians.

Acknowledgments

I would like to thank my academic advisor Kenneth Ribet for helping and supporting me both academically and financially for the past three years as a graduate student. Thank you for being always available and willing to meet, talk, and help with my work. I could not have hoped to have a better advisor and mentor.

I would also like to thank Kristin Lauter for inviting me to Microsoft Research and introducing me to genus 3 curves. I am incredibly thankful for her mentoring, for being available despite her busy schedule, for inviting me to visit La Jolla on several occasions, and for introducing me to numerous people in the cryptographic community. It has been a pleasure to work with her for the past two years.

Finally, I would like to thank my family, and in particular Megan, whose unwavering support through good and bad times has been invaluable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Discrete Logarithms

Given a finite cyclic group $\mathbb{G}$ with generator g, the *Discrete Logarithm Problem (DLP)* $\mathtt{DLog}_{\mathbb{G},g}(h)$ is the computational problem of writing an element $h \in \mathbb{G}$ in terms of g, i.e. finding an integer x such that $g^x = h$. This is obviously easy by simple enumeration if $\mathbb{G}$ is small, but for example if $\#\mathbb{G} > 2^{100}$ it is not clear how x could be found if $\mathbb{G}$ has no obviously useful extra structure. The idea then is to construct cryptographic primitives in such a way that their security is in some sense related to the hardness of solving a DLP in $\mathbb{G}$. Typically this means that if an attacker can solve a particular DLP quickly, then they can also break the related cryptosystem quickly. So-called *generic* (or *collision*, or *square-root*) *attacks* solve any DLP in $O(\#\mathbb{G}^{1/2})$ groups operations. Perhaps the most important such attack is *Pollard rho* and its distributed variations.

One standard choice is to take $\mathbb{G}$ to be a large prime order subgroup of $\mathbb{F}_p^\times$ for some large enough prime p. This was used by Diffie and Hellman in their seminal paper [DH], which is usually considered to have started the development of public key cryptography. The problem with $\mathbb{F}_p^\times$ is that certain algebraic structure of the integers can be exploited to mount a powerful attack with subexponential complexity called *index calculus* [Adl]. To ensure security, the prime p must be taken to be significantly larger than one might naively have expected. In more recent terminology *index calculus* is used to mean any algorithm with a structure similar to that of [Adl] to break the DLP in some group $\mathbb{G}$ with strong enough algebraic properties. Such generalized index calculus algorithms do not necessarily have subexponential complexity and may or may not be faster than Pollard rho.

Koblitz [Kob] and Miller [Mil] independently suggested using a large prime order subgroup of the group of $\mathbb{F}_q$-valued points on an elliptic curve $E/\mathbb{F}_q$. Note that $\#E(\mathbb{F}_q) = q + O(q^{1/2})$ according to Hasse's theorem. These suggestions turned out to be remarkably successful for several reasons. Generating such groups is easy, arithmetic is very efficient and perhaps most importantly when q is prime there is no efficient index calculus type algorithm known. The index calculus algorithm of Semaev [Sem] might in some cases perform better than generic methods, but it is generally not considered to be efficient enough to be a threat. The improvement [Sem2] works when q is a power of 2, but it is not yet clear if this is a significant threat.

Koblitz [Kob2] suggested using a prime order subgroup of the group of $\mathbb{F}_q$-valued points on the Jacobian variety Jac_C of a genus $g > 1$ hyperelliptic curve $C/\mathbb{F}_q$. The potential advantage over using elliptic curves is that the size of $\mathrm{Jac}_C(\mathbb{F}_q)$ is $\#\,\mathrm{Jac}_C(\mathbb{F}_q) = q^g + O(q^{(2g-1)/2})$ according to the Hasse-Weil theorem. Since the security level is in principle determined by the size of the group, if we take g to be larger, it suffices to use a smaller q to obtain a particular security level. Arithmetic on the Jacobian of a hyperelliptic curve can always be done reasonably fast using *Mumford coordinates* [Was]. When $g \geq 3$ most curves are non-hyperelliptic. These curves are less useful in cryptography due to the lack of efficient coordinate systems.

Background in cryptography, the theory of abelian varieties over finite fields, and the CM method is discussed in Chapter 2.

1.2 Index Calculus in Genus 3

The security and practicality of hyperelliptic cryptosystems turns out to depend hugely on the genus g. In genus 2 there are no known attacks faster than Pollard rho. The smaller field size and alternate models for the group make the arithmetic very efficient, even efficient enough to be competitive with elliptic curve cryptography (see [BCHL, BCLS]).

For curves of very large genus subexponential index calculus attacks were discovered by Adleman, DeMarrais, and Huang [ADH], and improved by Gaudry, Enge, and others [Gau, Eng, EG, VJS]. For small genus but still with $g > 3$, the expected security was drastically reduced by attacks, which, although exponential, were so much better than Pollard rho so as to render the complexity/security trade-off unacceptable (see e.g. [Gau, The, GTTD, Nag]).

The case of genus 3 has remained unclear and debated for a long time. The hyperelliptic locus of genus 3 curves is of codimension 1, which means that almost all genus 3 curves are non-hyperelliptic (Theorems 2.6.6, 2.6.7, 2.6.1). The non-hyperelliptic curves admit a smooth quartic plane embedding, coming from the canonical divisor [Har]. The rest of the genus 3 curves are hyperelliptic and non-planar of higher degree. In the hyperelliptic case a *double large prime index calculus* algorithm [GTTD] reduces the complexity of the DLP from $\widetilde{O}(q^{3/2})$ (with Pollard rho) to $\widetilde{O}(q^{4/3})$, where the notation $\widetilde{O}$ hides both constants and logarithmic factors. However, for practical field sizes this attack does not seem to be efficient enough to rule out the use of genus 3 hyperelliptic curves. On the other hand, the low degree geometry of non-hyperelliptic genus 3 curves was very cleverly exploited by Diem to yield a much more powerful double large prime index calculus attack against non-hyperelliptic genus 3 curves, with complexity $\widetilde{O}(q)$. In fact, Diem has been developing index calculus algorithms on Jacobians of low-degree plane curves [Die, Die2], of which non-hyperelliptic genus 3 curves are a special case [DT]. Both the hyperelliptic and the non-hyperelliptic double large prime index calculus attacks consist roughly speaking of three steps:

1) *Graph building*, where an extremely large graph with vertex set a subset of $C(\mathbb{F}_q)$ is built. The edges in the graph are labeled by certain linear relations among divisors on the curve modulo principal divisors.

2) *Relation collection*, where more linear relations are constructed and the graph of the previous step is used to reduce the relations to involve only elements in a much smaller subset of divisors.

3) *Linear algebra*, where a large sparse matrix is formed from the collected linear relations, for which a non-trivial kernel vector must found. The solution to the DLP can be read from this vector.

Even though the computational complexity of Diem's $\widetilde{O}(q)$ attack does seem impressive, one should keep in mind that the $\widetilde{O}$-notation can hide factors that are significant for practical field sizes and that it is crucial to understand how large these hidden factors are. But by far the biggest problem with both double large prime index calculus attacks is that they require immense amounts of memory to store the graph. As a result, the security of even the seemingly simpler non-hyperelliptic case has remained unclear.

In this thesis we develop, implement, and study both in theory and in practice a new kind of variant of Diem's $\widetilde{O}(q)$ index calculus against genus 3 non-hyperelliptic curves. This new attack improves the three steps listed above in several ways:

1) Graph building is extremely slow in the beginning. Our new approach targets precisely this problem and makes the graph grow almost instantaneously to a significant size, completely avoiding a long period of extremely slow growth. Moreover, this improvement removes the need to perform certain initialization work that was not explicitly mentioned earlier.

2) We combine relation collection and graph building into one step, performing them simultaneously. This improves the total running time of the algorithm.

3) The linear relations obtained using the new attack are simpler, thus yielding sparser linear system. This speeds up the linear algebra step mentioned above.

As a result, we obtain a surprisingly significant improvement to the running time of Diem's algorithm at the cost of a slight increase in total memory consumption. The performance of the new algorithm is well understood in theory (see Table 3.1 and Table 3.3), making all hidden constant and logarithmic factors explicit. Practical experiments yield running times closely matching the theoretical results (see Table 3.2). Detailed comparisons to Diem's algorithm and Pollard rho are presented in Figures 3.1 and 3.3.

Perhaps the biggest limitation of all hyperelliptic and non-hyperelliptic genus 3 double large prime index calculus algorithms is that they require massive amounts of memory to store the graph. We tackle this issue of memory consumption by studying a variant of our algorithm where memory consumption is restricted to a fraction of the original, and explain how this only slightly worsens the computational complexity (see Figure 3.2). We also study a parallelization scheme aiming to split the memory cost among several large memory storage units. The memory requirements of all these improvements are analyzed in great detail, making all hidden logarithmic factors explicit (for examples, see Tables 3.4, 3.5, 3.6). As a result the security of genus 3 non-hyperelliptic curves is now well understood, assuming our new index calculus attack is the best one available. Next we explain how this affects the security of hyperelliptic genus 3 curves.

Index calculus for Jacobians of non-hyperelliptic genus 3 curves is discussed in Chapter 3. All material in Chapter 3 is based on the paper [LL], which is joint work with Kristin Lauter.

1.3 Avoiding Isogeny Attacks in Genus 3

Although the above only applies to non-hyperelliptic genus 3 curves, it turns out to have a direct impact on the security of hyperelliptic genus 3 curves due to so-called *isogeny attacks*, as was first demonstrated by Smith [Smi]. Given a DLP on the Jacobian Jac_C of a non-hyperelliptic genus 3 curve $C/\mathbb{F}_q$, an attacker might be able to compute an isogeny to the Jacobian $\mathrm{Jac}_{C'}$ of another genus 3 curve $C'/\mathbb{F}_q$. Suppose this can be done explicitly enough to map the DLP to $\mathrm{Jac}_{C'}(\mathbb{F}_q)$. Since the locus of hyperelliptic curves in the moduli space of all genus 3 curves has codimension 1 (Theorems 2.6.6, 2.6.7, 2.6.1), C' will almost certainly be non-hyperelliptic unless the isogeny is chosen in some very special manner. The attacker can again employ the $\widetilde{O}(q)$ index calculus to break the cryptosystem, making hyperelliptic curves no more secure than non-hyperelliptic curves. Of course, this is the case only when such an isogeny can be explicitly computed. Luckily Smith's method does not work for all hyperelliptic genus 3 curves. A computation in [Smi] shows that approximately 18.57% of hyperelliptic curves can be expected to be vulnerable. Moreover, given a hyperelliptic curve it is easy to determine whether or not Smith's attack will work, so all that needs to be done to avoid it is to choose a curve such that necessary conditions are not met.

Unfortunately the situation is much more complicated than the above suggests. Recently there has been significant theoretical and practical progress in computing much more general isogenies than those used by Smith. In [LR, CR, Rob] the authors explain how to compute certain (ℓ, ℓ, ℓ)-isogenies (see Definition 4.1.3) and demonstrate in great detail how this can be done in the case of genus 2. At the time of writing this the (ℓ, ℓ, ℓ)-isogeny algorithm has only been implemented for genus 2 curves, but it is expected to work with appropriate modifications also for genus 3 curves.

The most general types of isogenies one might hope to compute are *cyclic isogenies*, i.e. isogenies with a cyclic kernel. This is expected to be significantly more difficult than computing (ℓ, ℓ, ℓ)-isogenies, mainly because cyclic isogenies do not respect principal polarizations and it is not clear how to recover an appropriate principal polarization on the target abelian variety. One possible approach to computing cyclic isogenies between Jacobians is explained in [Rob2] and further developed in the case of genus 2 by Dudeanu [Dud]. However, the method of Dudeanu places extremely strong restrictions on the rings of real multiplication of the source and the target Jacobians, and it is not clear if a similar type of approach could at all be used in the case of genus 3. Nevertheless, it is not unreasonable to assume that in the near future also cyclic isogenies between Jacobians of genus 3 curves can be explicitly computed.

This means that the security of genus 3 hyperelliptic curves, in particular for small field sizes (around 70 bits), depends crucially on whether or not an isogeny attack can be performed. In the last part of this thesis we explore possibilities for generating cryptographically interesting hyperelliptic genus 3 curves[1] with properties that make isogeny attacks particularly hard to perform. Our isogeny attack prevention methods are all based on the simple observation, used also in the genus 3 CM method of Weng, that if a simple 3-dimensional Jacobian over $\mathbb{F}_q$ contains an automorphism of order 4 then it must be the Jacobian of a genus 3 hyperelliptic curve (see [Wen] or Theorem 4.4.2). By very carefully choosing a sextic CM field $K/\mathbb{Q}$, a prime p and a p-Weil number $\pi \in \mathcal{O}_K$, and constructing the hyperelliptic curve using the genus 3 CM method, we can control the degrees of possible isogenies that change the endomorphism ring by removing the automorphism of order 4 from it. This is precisely what the attacker would have to do to reach the Jacobian of a non-hyperelliptic curve (see Theorem 4.4.2). Our construction prevents almost all maximal isotropic isogenies from being efficiently computable, and under some justifiable hypotheses about the computability of cyclic isogenies it prevents also those from working. The only reasonable approach left for the attacker is to compute a long enough chain of (ℓ, ℓ, ℓ)-isogenies with kernels stable under the action of the Frobenius. It is not clear how hard finding and computing such a chain is in a generic case, but it might be possible to further choose the hyperelliptic curve in a way that makes such chains impossible. This requires still more work and is left for a future project.

Avoiding isogeny attacks is discussed in Chapter 4.

[1]More precisely, we generate appropriate and realistic input data for the *genus* 3 *CM method* of Weng [Wen], which generates the curve.

Chapter 2

Background

2.1 Public Key Cryptography

We start by recalling the formal definition of a *public key encryption scheme*. We refer the reader to [KL] for more details, including formal definitions of realistic security models.

A public key encryption scheme consists of the following *probabilistic polynomial time* (often abbreviated PPT) algorithms:

> The algorithm $\texttt{KeyGen}$ takes as input a *security parameter* n and outputs a public key $\texttt{pk}$, a secret key $\texttt{sk}$, a *message space* $\mathcal{M}$ and a *ciphertext space* $\mathcal{C}$.
>
> The randomized algorithm $\texttt{Enc}$ takes as input the public key $\texttt{pk}$ and a message $m \in \mathcal{M}$, and outputs a ciphertext $\texttt{Enc}_{\texttt{pk}}(m) \in \mathcal{C}$.
>
> The deterministic algorithm $\texttt{Dec}$ takes as input the secret key $\texttt{sk}$ and a ciphertext $c \in \mathcal{C}$, and outputs a plaintext $\texttt{Dec}_{\texttt{sk}}(c) \in \mathcal{M}$ or $\texttt{FAIL}$.

Remark 2.1.1. In *private key encryption schemes* the security parameter is typically the key length. In public key schemes it is usually related to the key length in some more complicated way, depending on the scheme.

Definition 2.1.2 (Public key encryption scheme)**.** A public key encryption scheme consists of a triple $(\texttt{KeyGen}, \texttt{Enc}, \texttt{Dec})$ of algorithms as detailed above, such that

$$\Pr\left[\texttt{Dec}_{\texttt{sk}}(\texttt{Enc}_{\texttt{pk}}(m)) = m\right] \geq 1 - \operatorname{negl}(n)\,,$$

where the probability is taken over the output of $\texttt{KeyGen}$, the message space and the randomness used by $\texttt{Enc}$. Here $\operatorname{negl}(n)$ denotes a function such that $1/\operatorname{negl}(n)$ grows faster than any polynomial in n.

Such encryption schemes form the basis of secure communication over the internet, as they allow multiple clients to securely communicate with one host without having to deal with the problem of distributing and storing a private key for each client separately.

2.2 Discrete Logarithms

Typically public key encryption schemes are based on certain well-known and supposedly hard computational problems in algebra, geometry or number theory. Most often this means that if the computational problem can be solved fast, then also the cryptosystem can be broken fast. Unfortunately, the other direction is in many cases not known, i.e. even if we have a fast way of breaking the cryptosystem we might not know how to use it to solve the computational problem, but perhaps only some slightly easier less studied variant of it. Nevertheless, apparent hardness of the well-known computational problem is often thought to strongly suggest that the cryptosystem is in practice equally hard to break.

Although the need for indeterminacy in encryption (see Definition 2.1.2) complicates things somewhat, a vast number of different types of public key cryptosystems are known. In this work we focus on ones based on the following class of problems.

Definition 2.2.1 (Discrete Logarithm Problem (DLP))**.** Let $\mathbb{G}$ be a finite cyclic (abelian) group and $g \in \mathbb{G}$ a generator. Given an element $h \in \mathbb{G}$, the base-g *discrete logarithm* of h is an integer x (modulo the group order) such that $g^x = h$ (multiplicative notation). The *discrete logarithm problem* $\texttt{DLog}_{\mathbb{G},g}(h)$ is the problem of computing x given $(\mathbb{G}, g, h)$.

Of course the hardness of $\texttt{DLog}_{\mathbb{G},g}(h)$ depends very strongly on $\mathbb{G}$. For example, unless the group has large enough order one might be able to simply try all numbers between 0 and $\#\mathbb{G} - 1$ until the correct one is found. Also in certain groups the DLP is inherently easy unless the group order is impossibly huge.

Example 2.2.2. If $\mathbb{G} := \mathbb{Z}/p\mathbb{Z}$ (additive group), computing a discrete logarithm requires only the computation of an inverse of the generator and a multiplication modulo p. This means that the DLP can be solved by doing $O(\log_2 p)$ multiplications in integers modulo p, which is very easy unless p is extremely large.

Example 2.2.3. Perhaps the most classical choice for $\mathbb{G}$ is the multiplicative group $\mathbb{F}_q^\times$ of a finite field. Unfortunately such groups suffer from an attack (see Section 2.4) that forces q to be much larger than one might hope.

Example 2.2.4. An extremely successful choice for $\mathbb{G}$, independently suggested by Koblitz [Kob] and Miller [Mil], is the group of $\mathbb{F}_q$-valued points on an elliptic curve over $\mathbb{F}_q$. The advantage over the group $\mathbb{F}_q^\times$ is that the attack mentioned in Example 2.2.3 does not work, allowing q and hence the representatives of the group elements to be much smaller. In addition, elliptic curves provide algebraic geometric extra structure that has been used in clever ways to yield cryptosystems with a number of interesting properties.

Example 2.2.5. Koblitz [Kob2] suggested using the group of $\mathbb{F}_q$-valued points on the Jacobian of a hyperelliptic genus $g > 1$ curve over $\mathbb{F}_q$. The hardness of the DLP in such groups is a complicated topic and depends very strongly on the genus g. Most of this thesis is dedicated to understanding the case of $g = 3$.

We conclude this section by discussing perhaps the simplest and most famous DLP-based public key cryptosystem: The *Elgamal encryption scheme*, described by Taher Elgamal in [ElG]. In the Elgamal scheme $\mathbb{G}$ can be taken to be any finite cyclic group, although to achieve a reasonable level of security one should choose $\mathbb{G}$ carefully.

It should be pointed out that finding good groups to use is far from trivial. The groups should have large order, divisible by a large prime (due to the *Pohlig-Hellman decomposition*, which we discuss in Section 2.3) and as little extra mathematical structure as possible, while still admitting simple, compact representations for the group elements and very fast arithmetic. Preferably one would hope to have an easy-to-construct family of such groups and not just one group.

Remark 2.2.6. On the other hand, extra structure in the group might be used to construct cryptosystems with more interesting properties, a good example of which are the applications of supersymmetric elliptic curves to identity-based encryption.

Example 2.2.7 (Elgamal encryption scheme). Let $q := \#\mathbb{G}$. The secret key consists of a generator g of $\mathbb{G}$ and an integer x chosen uniformly at random from $\mathbb{Z}/q\mathbb{Z}$. The public key consists of the generator g and an element $h := g^x$. The message space is $\mathcal{M} := \mathbb{G}$ and the ciphertext space is $\mathcal{C} := (\mathbb{Z}/q\mathbb{Z}) \times \mathbb{G}$. To encrypt, choose an r from $\mathbb{Z}/q\mathbb{Z}$ uniformly at random. If m is the message, let

$$\texttt{Enc}_{\texttt{pk}}(m) := (g^r, m \cdot h^{-r}) \,.$$

To decrypt a ciphertext $(y, c) \in \mathcal{C}$, compute

$$\texttt{Dec}_{\texttt{sk}}(y, c) := c \cdot y^x \,.$$

If an adversary is able to compute discrete logarithms easily in the group $\mathbb{G}$, then they can find r from y and compute $m = c \cdot h^r$. Much can be said about the security of generic Elgamal, i.e. without specifying what the group $\mathbb{G}$ is, for which we refer the reader to [KL] since our goals lie elsewhere.

2.3 Generic Attacks

Next we take a look at some well-known attacks for solving $\texttt{DLog}_{\mathbb{G},g}(h)$. These attacks are generic in the sense that they work for any group $\mathbb{G}$. We will later see that in practice all groups that are used as $\mathbb{G}$ have a strong algebraic or geometric structure, which can in many cases be exploited to yield much more powerful attacks.

Pohlig-Hellman Decomposition

It turns out to be surprisingly difficult to choose a secure group $\mathbb{G}$ as we briefly discussed before Remark 2.2.6. Theorem 2.3.1 explains why $\mathbb{G}$ should be such that $\#\mathbb{G}$ is divisible by a large enough prime.

Theorem 2.3.1 (Pohlig-Hellman)**.** *If the group order q is a product of powers of small primes p_i,*

$$q = p_1^{e_1} \dots p_k^{e_k} \,,$$

then solving $\mathtt{DLog}_{\mathbb{G},g}(h)$ is easy if solving discrete logarithms in all subgroups of $\mathbb{G}$ of prime order is easy.

Proof. The proof of Theorem 2.3.1 is a classical application of the Chinese Remainder Theorem. Denote

$$g_i := g^{q/p_i^{e_i}} \,, \qquad h_i := h^{q/p_i^{e_i}} \,, \qquad x_i :\equiv x \pmod{p_i^{e_i}} \,.$$

Then $h_i = g_i^{x_i}$. Let $\mathbb{G}_i$ denote the subgroup of order $p_i^{e_i}$. If we can solve all $\mathtt{DLog}_{\mathbb{G}_i,g_i}(h_i)$ we can piece together the x_i using the Chinese Remainder Theorem to find $x \pmod{q}$.

It remains to show how to solve $\mathtt{DLog}_{\mathbb{G}_i,g_i}(h_i)$ when p_i is small. Since x_i is defined modulo $p_i^{e_i}$, we can write

$$x_i = \alpha_{i,0} + \alpha_{i,1} p_i + \ldots + \alpha_{i,e_i-1} p_i^{e_i-1} \,,$$

so

$$h_i = g_i^{\alpha_{i,0}} \left(g_i^{p_i}\right)^{\alpha_{i,1}} \dots \left(g_i^{p_i^{e_i-1}}\right)^{\alpha_{i,e_i-1}} \,.$$

Raising this to the $p_i^{e_i-1}$-th power yields the equation

$$h_i^{p_i^{e_i-1}} = \left(g_i^{p_i^{e_i-1}}\right)^{\alpha_{i,0}} \,.$$

This is a DLP in the subgroup of $\mathbb{G}$ of order p_i which we assume we can solve, i.e. we assume that the $\alpha_{i,0}$ can be found. Next compute

$$\left(h_i g_i^{-\alpha_{i,0}}\right)^{p_i^{e_i-2}} = \left(g_i^{p_i^{e_i-1}}\right)^{\alpha_{i,1}}$$

and again suppose we can solve this DLP to recover the $\alpha_{i,1}$. We can continue in this way to recover all $\alpha_{i,j}$ and eventually all x_i. □

Theorem 2.3.1 implies that we really need $\mathbb{G}$ to contain at least one subgroup where the DLP is hard to solve. In other words, solving a DLP in $\mathbb{G}$ is as hard as solving a DLP in the largest prime order subgroup of $\mathbb{G}$. In practice this means that $\#\mathbb{G}$ should be divisible by a large prime, since the security usually always depends most critically on the order of the group.

Pollard Rho

Let $q := \#\mathbb{G}$. It is clear that $\mathtt{DLog}_{\mathbb{G},g}(h)$ can always be solved in time $O(q)$ by exhaustive search, but even in generic groups the DLP can actually be solved much faster using a *collision algorithm.*

We now describe the *Pollard rho* algorithm [Pol], which is a *generic collision/square-root algorithm* to solve $\mathtt{DLog}_{\mathbb{G},g}(h)$, meaning it works for any group $\mathbb{G}$ and has complexity $O\left(q^{1/2}\right)$ instead of $O(q)$. Imagine choosing random elements $x_1, x_2, \ldots$ from $\mathbb{Z}/q\mathbb{Z}$ and constructing a random walk $g^{x_1}, g^{x_2}, \ldots$ until we hit h. This can be expected to terminate successfully in time $O(q)$. Now suppose instead that we store all of the elements that we have already walked through and at some point observe a collision in the walk. If we set up the random walk in a particular way, this collision is enough to solve the DLP and one can show that finding such a collision is significantly easier than hitting the exact element h.

We follow [Gal2] where many more details and improvements can be found. The idea is to find integers $a_i, b_i, a_j, b_j \in \mathbb{Z}/q\mathbb{Z}$, $b_i \neq b_j$, such that

$$g^{a_i}h^{b_i} = g^{a_j}h^{b_j} ,$$

from which we can solve

$$h = g^{(a_i-a_j)(b_j-b_i)^{-1} \pmod q}$$

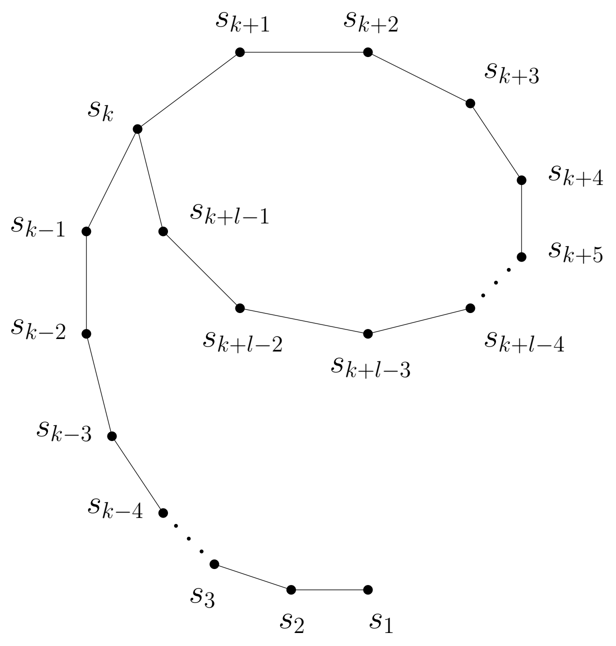
and so find the discrete logarithm.

The way to do this is to randomly generate a sequence of elements of the form $s_i := g^{a_i}h^{b_i}$ until a match is found, which happens faster than "expected", which is explained by the birthday paradox. Of course it is crucial to know the exponents a_i, b_i for every s_i in this sequence. The classical way of doing this is to define a function $f : \mathbb{G} \to \mathbb{G}$ which is pseudorandom enough and explicit in how it changes the exponents a_i, b_i. The sequence (s_i) is then generated iteratively by choosing $s_1 = g$ and setting $s_{i+1} = f(s_i)$. Because $\mathbb{G}$ is finite, a collision will eventually happen, i.e. $s_k = s_{k+l}$ for some k and $l > 0$. This is depicted in Figure 2.1. Note also that after the collision the values of the sequence will keep repeating in a cycle of length l. It is easy to find a function f that is pseudorandom enough and has the required properties. We refer the reader to [Gal] for details.

The main difficulty is to recognize that a collision has occurred. Naively one might imagine that all s_i need to be stored and compared to every new value in the sequence to detect a collision, which would require a vast amount of time and memory. The method used by Pollard in [Pol] was, instead of just computing the sequence (s_i), to compute both (s_i) and (s_{2i}) simultaneously. The latter can be computed iteratively by starting with $s_2 = f(s_1)$ and always applying f twice since $s_{2(i+1)} = f(f(s_{2i}))$. A collision has occurred if $s_i = s_{2i}$. It is easy to see that this happens precisely when $i \geq k$ and $l \mid i$. Moreover, $s_i = s_{2i}$ occurs for some i between k and $k+l-1$ because there is precisely one number between k and $k+l-1$ that is divisible by l.

To say something about the complexity of this attack, one needs to know that the expected lengths of the tail and the cycle are $\sqrt{\pi q/8}$. In fact, the expected value of i for which $s_i = s_{2i}$ is conjectured to be approximately $0.823\sqrt{\pi q/2}$. Again, the reader is referred to [Gal] for details, but this proves that we can expect to be able to solve the $\mathtt{DLog}_{\mathbb{G},g}(h)$ in time $O\left(q^{1/2}\right)$.

Figure 2.1: Collision in the pseudorandom sequence



Example 2.3.2. At the time of writing this 2^{80} operations is close to the boundary of being possible to do, so a cryptosystem with 80 bits of security[1] might be possible to break with relatively modest algorithmic improvements, or improvements to CPU and memory technology. If Pollard rho is the best known attack against $\texttt{DLog}_{\mathbb{G},g}(h)$, then such a security level is obtained when $\#\mathbb{G}$ is divisible by a prime of size roughly 160 bits due to the Pohlig-Hellman decomposition (Theorem 2.3.1).

2.4 Index Calculus

In this section we discuss (classical) *index calculus*, which is an algorithm for solving the DLP when $\mathbb{G}$ is the multiplicative group of a finite field. Here we consider the field to be a prime field $\mathbb{F}_p$ so that $\mathbb{G}$ has $p-1$ elements, represented by integers in $[1, p-1]$. Index calculus is significantly different from e.g. Pollard rho and other generic methods in that it has a subexponential running time, but to achieve this it needs to use certain algebraic properties of $\mathbb{F}_p^\times$ that more general finite cyclic groups can not be expected to have. We present only a very classical version of index calculus here.

Suppose we have a discrete logarithm problem $h \equiv g^x \pmod p$. Let $\mathcal{F}_B$ be the set of all primes $\leq B$. We call this set the *factor base*. Suppose we know the base-g discrete

[1]A cryptosystem, or cryptographic primitive, is said to have b bits of security if it can be broken in 2^b operations with currently known methods.

logarithms of all $r \in \mathcal{F}_B$, i.e. numbers $\log_g r$ modulo $p-1$ such that $r \equiv g^{\log_g r} \pmod{p}$. If h is a product of powers of primes in $\mathcal{F}_B$ so that

$$h = \prod_{r \in \mathcal{F}_B} r^{e_r}\,, \tag{2.4.1}$$

then

$$x \equiv \sum_{r \in \mathcal{F}_B} e_r \log_g r \pmod{p-1}\,.$$

Of course it is extremely unlikely that h would be of the form (2.4.1) unless B is huge. The standard approach is to instead look for products $hg^k \pmod{p}$ that are of the form

$$hg^k = \prod_{r \in \mathcal{F}_B} r^{e_r}\,. \tag{2.4.2}$$

Once such an $hg^k \pmod{p}$ is found, x can be solved as

$$x \equiv \sum_{r \in \mathcal{F}_B} e_r \log_g r - k \pmod{p-1}\,.$$

The difficulty of finding an integer k such that (2.4.2) holds clearly depends extremely strongly on B.

It remains to explain how the logarithms $\log_g r$ can be computed. The standard approach is to choose random integers $l \in \mathbb{Z}/(p-1)\mathbb{Z}$, compute $g^l \pmod{p}$ and factor the result. If the prime decomposition involves only powers of primes in $\mathcal{F}_B$, then

$$g^l = \prod_{r \in \mathcal{F}_B} r^{e_r}\,, \qquad l = \sum_{r \in \mathcal{F}_B} e_r \log_g r\,.$$

Collecting $\#\mathcal{F}_B + 1$ such relations

$$\begin{aligned} l_1 &\equiv \sum_{r \in \mathcal{F}_B} e_{r,1} \log_g r \pmod{p-1} \\ l_2 &\equiv \sum_{r \in \mathcal{F}_B} e_{r,2} \log_g r \pmod{p-1} \\ &\vdots \qquad\qquad \vdots \\ l_{\#\mathcal{F}_B+1} &\equiv \sum_{r \in \mathcal{F}_B} e_{r,\#\mathcal{F}_B+1} \log_g r \pmod{p-1} \end{aligned}$$

yields an overdetermined sparse linear system in $\#\mathcal{F}_B$ variables $\{\log_g r\}_{r \in \mathcal{F}_B}$. This almost certainly has only one non-trivial solution, which can be found roughly in time $O\left(w \cdot (\#\mathcal{F}_B)^2\right)$ using sparse linear algebra techniques, where w is the average number of non-zero monomials in the equations.

Remark 2.4.1. Note that in the above example linear algebra must be done over $\mathbb{Z}/(p-1)\mathbb{Z}$, which complicates things a little bit. One might need to solve the equations modulo prime power factors of $p-1$ first and then combine these using the Chinese Remainder Theorem.

Finding one linear relation between the $\{\log_g r\}_{r\in\mathcal{F}_B}$ is harder when $\#\mathcal{F}_B$ is smaller, but on the other hand the number of relations needed is also smaller. When $\#\mathcal{F}_B$ is bigger the size of the linear system is bigger and the hardness of solving it increases quadratically. The usual approach is to choose B to be such that the complexity of the relation search step is of the same order as the complexity of the linear algebra step. One more aspect in the complexity consideration is the difficulty of efficiently determining whether a particular number modulo p actually decomposes as a product of powers of primes in $\mathcal{F}_B$. We will not go into such details and instead only present the well-known heuristic complexity result.

Theorem 2.4.2. *If B is chosen appropriately, the complexity of the index calculus algorithm can be made subexponential. It is possible to prove under some heuristic assumptions that a complexity of*

$$O\left(L_p(1/3,c)\right) = O\left(e^{(c+o(1))(\ln p)^{1/3}(\ln\ln p)^{2/3}}\right)$$

can be obtained, where c is some small constant.

Proof. See for example [SWD]. □

Example 2.4.3. Recall from Example 2.3.2 that if Pollard rho was the best attack, p would have to be at least 160 bits to ensure 80 bits of security. Unfortunately, due to index calculus the DLP in $\mathbb{F}_p^\times$ with such a small p is very easy to break. Instead one must take p to be around 1248 bits to ensure 80 bits of security. In practice p would often be taken to be 2048 bits.

The idea behind index calculus can be generalized to other groups with enough "good" structure. Let $\mathbb{G}$ be a cyclic group of prime order generated by g and consider the DLP $h = g^x$. We need

1. a subset $\mathcal{F} \subseteq \mathbb{G}$ called the factor base such that a large enough fraction of elements of $\mathbb{G}$ can be written as products of factor base elements;

2. a way of easily determining whether an element of $\mathbb{G}$ is a product of factor base elements and in the affirmative case finding the decomposition (in particular, we need a way of easily determining whether an element of $\mathbb{G}$ is in $\mathcal{F}$);

3. a way of easily constructing more than $\#\mathcal{F}$ non-trivial sparse relations involving only factor base elements.

If these conditions hold, the standard approach is to compute the complexity of finding $\#\mathcal{F}+1$ relations and balancing that with the complexity of solving a sparse linear system of $\#\mathcal{F}+1$ equations with some average weight[2] estimated as a function of $\#\mathcal{F}$. From this one can then solve the required $\#\mathcal{F}$.

There exists a number of improvements and modifications of the basic index calculus method. We will discuss later so-called *double large prime index calculus* algorithms, where the decompositions of group elements as in (2.4.2) is allowed to involve also up to two non-factor base group elements which are usually called *large primes*. In the classical $\mathbb{F}_p^\times$-version of double large prime index calculus the product is allowed to contain powers of at most two primes larger than the bound B, hence the terminology. These non-factor base elements must then somehow be eliminated to produce the linear system in $\#\mathcal{F}$ variables. The index calculus algorithms discussed in Chapter 3 are all of this type.

2.5 Abelian Varieties over Finite Fields

In the rest of this thesis we will mainly be interested in DLPs where the group $\mathbb{G}$ (recall Definition 2.2.1) is the group of $\mathbb{F}_p$-valued points on the Jacobian of a genus g curve. In the case of elliptic curves ($g=1$) this is known to yield (apparently) extremely secure and high performing cryptosystems, as was first suggested by Koblitz [Kob] and Miller [Mil]. Later Koblitz [Kob2] suggested using the Jacobians of hyperelliptic genus $g>1$ curves over $\mathbb{F}_p$. These are principally polarized abelian varieties where arithmetic is convenient to do using *Mumford coordinates* and *Cantor's algorithm* [Was, Can2].

In this Section we recall some basic properties of abelian varieties over finite fields and their isogeny classes.

Homomorphisms and Isogenies

Let $\mathcal{A},\mathcal{B}$ be abelian varieties over a finite field of characteristic p. We denote the $\mathbb{Z}$-module of homomorphisms $\mathcal{A}\to\mathcal{B}$ by $\operatorname{Hom}(\mathcal{A},\mathcal{B})$ and the ring of endomorphisms $\mathcal{A}\to\mathcal{A}$ by $\operatorname{End}\mathcal{A}$. It is often more convenient to instead work with the vector spaces

$$\operatorname{Hom}^0(\mathcal{A},\mathcal{B}) := \operatorname{Hom}(\mathcal{A},\mathcal{B})\otimes_{\mathbb{Z}}\mathbb{Q}\,, \qquad \operatorname{End}^0\mathcal{A} := \operatorname{End}\mathcal{A}\otimes_{\mathbb{Z}}\mathbb{Q}\,.$$

Definition 2.5.1. A homomorphism $f:\mathcal{A}\to\mathcal{B}$ is called an *isogeny* if $\operatorname{im} f=\mathcal{B}$ and $\ker f$ is finite. If an isogeny $\mathcal{A}\to\mathcal{B}$ exists we say that $\mathcal{A}$ and $\mathcal{B}$ are isogenous.

The *degree* of an isogeny $f:\mathcal{A}\to\mathcal{B}$ is the degree of the function field extension

$$\deg f := [K(\mathcal{A}) : f^*K(\mathcal{B})]\,.$$

In general, if the extension has separable degree $[K(\mathcal{A}) : f^*K(\mathcal{B})]_{\mathrm{sep}}$, then

$$\#(\ker f) = [K(\mathcal{A}) : f^*K(\mathcal{B})]_{\mathrm{sep}}\,.$$

[2]By the weight of a linear equation we mean the number of non-zero terms appearing in it.

If the extension is separable we call f a *separable isogeny*, in which case

$$\#(\ker f) = \deg f\,.$$

For more details see [Mum2].

Next we need to understand to what extent a finite subgroup $\mathcal{K} \subset \mathcal{A}$ determines an isogeny.

Theorem 2.5.2 ([Mum2])**.** *Let X be an algebraic variety and G a finite group of automorphisms of X. Suppose that for any $x \in X$, the orbit Gx of x is contained in an affine open subset of X. Then there is a pair (Y, g) where Y is a variety and $g : X \to Y$ a morphism such that*

i) as a topological space (Y, g) is the quotient of X for the G-action;

ii) if $g_*(\mathcal{O}_X)^G$ denotes the subsheaf of G-invariants of $g_*(\mathcal{O}_X)$ for the action of G on $g_*(\mathcal{O}_X)$ deduced from i), then the natural homomorphism $\mathcal{O}_Y \to g_*(\mathcal{O}_X)^G$ is an isomorphism.

The pair (Y, g) is determined up to isomorphism by these conditions. Moreover, g is finite, surjective and separable. The variety Y is called the quotient of X by G *and is denoted by X/G.*

If $X = \mathcal{A}$ is an abelian variety and $\mathcal{K} \subset \mathcal{A}$ a finite subgroup acting by translation, then $\mathcal{A}/\mathcal{K}$ admits a natural structure of an abelian variety.

Theorem 2.5.3 ([Mum2])**.** *Let $\mathcal{A}$ be an abelian variety. Then there is a one-to-one correspondence between the two sets of objects:*

1) finite subgroups $\mathcal{K} \subset \mathcal{A}$;

2) separable isogenies $f : \mathcal{A} \to \mathcal{B}$ where $\mathcal{B}$ is determined up to isomorphism.

The correspondence is set up by $\mathcal{K} = \ker f$ and $\mathcal{B} = \mathcal{A}/\mathcal{K}$.

Unfortunately, Theorem 2.5.3 does not describe all isogenies but only the separable ones. For a more complete description one needs to use the scheme theoretic approach and define quotients of group schemes, which we will not discuss but instead refer the reader to [Mum2].

One important example of a separable endomorphism is given by the multiplication-by-n map

$$[n] : \mathcal{A} \to \mathcal{A}\,.$$

The kernel of $[n]$, i.e. the group of n-torsion points of $\mathcal{A}$, is denoted $\mathcal{A}[n]$.

Theorem 2.5.4 ([Mum2])**.** *If $\dim \mathcal{A} = g$, then $\deg[n] = n^{2g}$. Let p be the characteristic of the field of definition of $\mathcal{A}$.*

If $p \nmid n$, then $[n]$ is separable, $\#\mathcal{A}[n] = n^{2g}$ and $\mathcal{A}[n] \cong (\mathbb{Z}/n\mathbb{Z})^{2g}$.

If $p \mid n$, then $[n]$ is inseparable. Moreover, there is an integer i with $0 \le i \le g$ such that $\mathcal{A}[p^m] \cong (\mathbb{Z}/p^m\mathbb{Z})^i$ for all $m \ge 1$.

Definition 2.5.5. An abelian variety $\mathcal{A}$ is called *ordinary* if $i = g$ in Theorem 2.5.4.

Lemma 2.5.6. *If there is an isogeny $f : \mathcal{A} \to \mathcal{B}$, then there is also an isogeny $\widehat{f} : \mathcal{B} \to \mathcal{A}$ such that*

$$f \circ \widehat{f} = \widehat{f} \circ f = [\deg f]\,.$$

Proof. The reason is that $\ker f \subseteq \mathcal{A}[\deg f]$ so $[\deg f] : \mathcal{A} \to \mathcal{A}$ factors through $\mathcal{A}/\ker f$. □

Lemma 2.5.6 implies that it is reasonable to talk about two abelian varieties being isogenous. In fact, being isogenous is an equivalence relation and we can talk about the isogeny class of an abelian variety.

Remark 2.5.7. Note that $\widehat{f}$ is not what is usually called the *dual isogeny*.

Definition 2.5.8. Let $\mathcal{A}$ be defined over $\mathbb{F}_q$ and let $\pi \in \mathrm{Gal}(\overline{\mathbb{F}}_q/\mathbb{F}_q)$ be the Frobenius automorphism. Since π leaves $\mathcal{A}$ invariant, it induces an endomorphism $\pi \in \mathrm{End}\,\mathcal{A}$ called the *Frobenius endomorphism*.

Clearly the Frobenius $\pi : \mathcal{A} \to \mathcal{A}$ acts as the identity precisely on the points $\mathcal{A}(\mathbb{F}_q)$. The degree of the Frobenius is

$$\deg \pi = [K(\mathcal{A}) : \pi^* K(\mathcal{A})] = q^g\,.$$

The extension is purely inseparable and $\ker \pi = \{0\}$, so π is an isogeny.

Honda-Tate Theory

Let $\mathcal{A}$ be a g-dimensional abelian variety over $\mathbb{F}_p$. In this section we recall how $\mathrm{End}\,\mathcal{A}$ and $\mathrm{End}^0\,\mathcal{A}$ reflect the structure of $\mathcal{A}$. In particular we present the famous result of Honda and Tate which classifies isogeny classes of abelian varieties over finite fields in terms of the endomorphism structure.

We start with a simple observation that tells us that the best we can possibly get from $\mathrm{End}^0\,\mathcal{A}$ is information about the isogeny class.

Lemma 2.5.9. $\mathrm{End}^0\,\mathcal{A}$ *only depends on the isogeny class of $\mathcal{A}$.*

Proof. Let $f : \mathcal{A} \to \mathcal{B}$ be an isogeny, $n := \deg f$, and $\widehat{f} : \mathcal{B} \to \mathcal{A}$ an isogeny such that $\widehat{f} \circ f = [n]$ (see Lemma 2.5.6). Then for any $\varphi \in \mathrm{End}^0\,\mathcal{A}$, $f \circ \varphi \circ n^{-1}\widehat{f} \in \mathrm{End}^0\,\mathcal{B}$. This yields a ring isomorphism $\mathrm{End}^0\,\mathcal{A} \cong \mathrm{End}^0\,\mathcal{B}$. □

Definition 2.5.10. An abelian variety is *simple* if it contains no non-trivial abelian subvarieties.

Theorem 2.5.11 (Poincaré-Weil [Mum2])**.** *Let $\mathcal{A}$ be an abelian variety. Then $\mathcal{A}$ is isogenous to*

$$\mathcal{A}_1^{n_1} \times \mathcal{A}_2^{n_2} \times \ldots \times \mathcal{A}_k^{n_k}\,,$$

where the abelian varieties $\mathcal{A}_i$ are simple and non-isogenous. The integers n_i are uniquely determined and $\mathcal{A}_i$ are uniquely determined up to isogeny.

Lemma 2.5.12. *If $\mathcal{A}$ is simple, every endomorphism is an isogeny.*

Proof. If $f \in \operatorname{End}\mathcal{A}$, then $\operatorname{im} f$ is an abelian subvariety of $\mathcal{A}$ so unless $f = 0$, $\operatorname{im} f = \mathcal{A}$ which also implies that $\ker f$ is finite, so f is an isogeny. □

Theorem 2.5.13 ([Mum2])**.** $\operatorname{End}^0 \mathcal{A}$ *is a finite-dimensional semisimple algebra over* $\mathbb{Q}$. *The Artin-Wedderburn theorem implies that such an algebra is a finite product of matrix algebras over division rings. If $\mathcal{A}$ is simple,* $\operatorname{End}^0 \mathcal{A}$ *is a division ring.*

The next theorem explains the structure of the endomorphism ring $\operatorname{End}^0 \mathcal{A}$ for simple abelian varieties and the significance of the Frobenius endomorphism.

Theorem 2.5.14 ([MW])**.** *Let $\mathcal{A}$ be a simple g-dimensional abelian variety over $\mathbb{F}_q$. Denote the characteristic polynomial of the Frobenius endomorphism by f, so $f(\pi)$ acts as* $[0]$ *on $\mathcal{A}$. Then*

1) $f = m^e$ for some integer e and irreducible monic polynomial m with integer coefficients;

2) the center of the division ring $\operatorname{End}^0 \mathcal{A}$ *is the number field $\mathbb{Q}(\pi)$;*

3) $[\operatorname{End}^0 \mathcal{A} : \mathbb{Q}] = e^2[\mathbb{Q}(\pi) : \mathbb{Q}]$ *and* $2g = e[\mathbb{Q}(\pi) : \mathbb{Q}]$*, so*

$$2g = [\operatorname{End}^0 \mathcal{A} : \mathbb{Q}(\pi)]^{1/2} \cdot [\mathbb{Q}(\pi) : \mathbb{Q}] \,;$$

4) the absolute value of π is $q^{1/2}$ under every embedding of $\mathbb{Q}(\pi)$ in $\mathbb{C}$. In other words, all roots of f have absolute value $q^{1/2}$.

Theorem 2.5.15 ([MW])**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be abelian varieties over a finite field $\mathbb{F}_q$ and $f_{\mathcal{A}}$, $f_{\mathcal{B}}$ the characteristic polynomials of their Frobenius endomorphisms. Then the following are equivalent:*

1) $\mathcal{A}$ and $\mathcal{B}$ are isogenous;

2) $f_{\mathcal{A}} = f_{\mathcal{B}}$;

3) The zeta-functions of $\mathcal{A}$ and $\mathcal{B}$ are the same, so $\#\mathcal{A}(\mathbb{F}_{q^n}) = \#\mathcal{B}(\mathbb{F}_{q^n})$ for every positive integer n.

Definition 2.5.16. An algebraic number π is called a q-Weil number if all of its complex embeddings have absolute value $q^{1/2}$.

Combining Theorems 2.5.14,2.5.15 yields the famous result of Honda and Tate.

Theorem 2.5.17 (Honda-Tate)**.** *There is one-to-one correspondence between isogeny classes of abelian varieties over a finite field $\mathbb{F}_q$ and conjugacy classes of q-Weil numbers.*

Definition 2.5.18. A *CM field* is a totally imaginary extension of a purely real number field.

Definition 2.5.19. A simple abelian variety $\mathcal{A}$ of dimension g over any field is said to be *of CM type* if $\operatorname{End}^0 \mathcal{A}$ is a *CM field* of degree $2g$.

If $\mathcal{A}$ in Definition 2.5.19 is defined over a finite field, then Theorem 2.5.14 implies that $\operatorname{End}^0 \mathcal{A}$ is generated as an algebra over $\mathbb{Q}$ by the Frobenius endomorphism. The endomorphism ring $\operatorname{End} \mathcal{A}$ is an order $\mathcal{O} \subseteq \mathcal{O}_K$ containing $\mathbb{Z}[\pi, \overline{\pi}]$.

2.6 Constructing Curves for Cryptography

Our goal is to understand the *CM method* for generating hyperelliptic genus 3 curves $C/\mathbb{F}_p$ such that the group of $\mathbb{F}_p$-valued points on its Jacobian $\operatorname{Jac}_C(\mathbb{F}_p)$ has a known almost prime order, i.e. the order is the product of a large prime and a small *cofactor*. We start by discussing an analogous result for elliptic curves which is much easier to understand. It was developed by Atkin and Morain in [AM] for their elliptic curve primality testing algorithm.

Elliptic Curves

Let E be an elliptic curve over complex numbers defined by an equation $y^2 = x^3 + Ax + B$, where $A, B \in \mathbb{C}$. The discriminant $\Delta := -4A^3 - 27B^2$ must be non-zero for this equation to define a non-singular plane curve. The j-invariant

$$j(E) := 1728 \frac{4A^3}{4A^3 + 27B^2}$$

is well-known to classify isomorphism classes of elliptic curves over algebraically closed fields (see [Was, Sil1]), so the moduli space of elliptic curves up to isomorphism is parametrized by the j-invariant. Recovering an elliptic curve (or a representative for the isomorphism class) from a given j-invariant is also easy, namely the curve

$$y^2 = x^3 - \frac{27j}{4(j-1728)}x + \frac{27j}{4(j-1728)} \tag{2.6.1}$$

has j-invariant j.

Theorem 2.6.1 (Torelli [Mil2])**.** *Over an algebraically closed field the Jacobian variety of a genus g curve (up to isomorphism) is uniquely determined up to isomorphism as a principally polarized g-dimensional abelian variety.*

According to Theorem 2.6.1, to understand isomorphism classes of elliptic curves we can just as well study principally polarized 1-dimensional abelian varieties[3]. Over complex

[3]Of course elliptic curves are isomorphic to their Jacobians, but this is an instructive point of view that helps us understand the higher dimensional case.

numbers these are precisely the compact Riemann surfaces $\mathbb{C}/\Lambda$, where $\Lambda_\tau := \mathbb{Z} + \tau\mathbb{Z}$ is the period lattice and $\Im\tau > 0$. In the complex upper half-plane two values τ and τ' yield the same lattice precisely when they differ by a Möbius transformation.

Let $q := e^{2\pi i\tau}$ where τ is in the complex upper half-plane, and let $\sigma_3(n)$ denote the sum of cubes of divisors of an integer n. On the complex upper half-plane the function

$$\widetilde{j}(\tau) := \frac{\left(1 + 240\sum_{n=1}^{\infty}\sigma_3(n)q^n\right)^3}{q\prod_{n=1}^{\infty}\left(1-q^n\right)^{24}} \tag{2.6.2}$$

is invariant under Möbius transformations and one can show (see [Sil2]) that the field of all such functions is $\mathbb{C}(\widetilde{j})$. Since the modular invariant $\widetilde{j}$ is a function on the moduli space of these 1-dimensional principally polarized abelian varieties over $\mathbb{C}$ (i.e. the space of complex 1-dimensional algebraic tori), it should be possible according to Theorem 2.6.1 to relate it to the classical invariant j on the moduli space of genus 1 curves. Indeed, the correspondence is exactly what one would hope.

Theorem 2.6.2 ([Sil1])**.** *For an elliptic curve E and the corresponding complex torus $\mathbb{C}/\Lambda_\tau$,*

$$j(E) = \widetilde{j}(\tau)\,.$$

Let Λ be the lattice corresponding to an elliptic curve $E/\mathbb{C}$, in which case the endomorphism ring is simply

$$\operatorname{End} E = \{\alpha \in \mathbb{C} \mid \alpha\Lambda \subseteq \Lambda\}\,.$$

It is not hard to show that $\operatorname{End} E$ is either $\mathbb{Z}$ or an order $\mathcal{O}$ in an imaginary quadratic field (a CM field) K, i.e. a full sublattice of $\mathcal{O}_K$ which is also a subring.

Theorem 2.6.3 ([Sil2])**.** *Let $\iota : K \hookrightarrow \mathbb{C}$ be an embedding of the imaginary quadratic field K in $\mathbb{C}$. Then the lattice Λ equals $\iota(\mathfrak{a})$ for some ideal $\mathfrak{a} \subseteq \mathcal{O}$.*

Moreover, $\mathfrak{a}$ is determined up to principal fractional $\mathcal{O}$-ideals so the lattice is determined up to scaling by an element of the ring class group $\mathrm{Cl}(\mathcal{O})$. This establishes a one-to-one correspondence between the set of isogenous elliptic curves up to isomorphism and the ring class group $\mathrm{Cl}(\mathcal{O})$.

We suppose from now on that $\operatorname{End} E = \mathcal{O}_K$[4]. The numbers $j(E)$ have some amazing properties.

Theorem 2.6.4 ([Sil2])**.** *For any elliptic curve E, $j(E)$ is an algebraic integer over $\mathbb{Q}$ and conjugate j-invariants belong to isogenous elliptic curves.*

From Theorem 2.6.3 and Theorem 2.6.4 it follows that the minimal polynomial of $j(E)$ has degree equal to the class number h_K of K and $\mathbb{Q}(j(E))$ is the Hilbert class field of K [Sil2].

Computing representatives for the entire isogeny class of a given elliptic curve $E = \mathbb{C}/\iota(\mathfrak{a})$ with CM by $\mathcal{O}_K$ ($\mathfrak{a}$ is an $\mathcal{O}_K$-ideal) can be done as follows. Fix an embedding $\iota : K \hookrightarrow \mathbb{C}$.

[4]This is merely for the sake of simplicity.

Find representatives $\mathfrak{a}_i$ for each ideal class and for the ideal lattices $\iota(\mathfrak{a}_i)$ compute the modular j-invariants $\widetilde{j}(\iota(\mathfrak{a}_i))$ using (2.6.2) up to high enough precision. Then form the polynomial

$$H_K(x) := \prod_{i=1}^{h_K} \left(x - \widetilde{j}(\iota(\mathfrak{a}_i)) \right) \tag{2.6.3}$$

and recognize the coefficients of the polynomial as integers. Note that each elliptic curve over $\mathbb{C}$ in the isogeny class can be found from the conjugate roots of this polynomial using (2.6.1).

In cryptography we are not interested in elliptic curves over complex numbers but rather over some finite field $\mathbb{F}_p$. We restrict to prime fields here. An elliptic curve $E/\mathbb{F}_p$ has $\operatorname{End} E$ either an order $\mathcal{O}$ in an imaginary quadratic field K or an order in some (non-commutative) quaternion algebra over $\mathbb{Q}$. In the former case E is ordinary as an abelian variety, and this is the only case we consider from now on.

Remark 2.6.5. It is easy to see that $\operatorname{End} E$ must be strictly bigger than $\mathbb{Z}$ because the (inseparable) Frobenius endomorphism always exists and does not correspond to multiplication by any integer.

Suppose $E/\mathbb{C}$ has a local model over $\mathbb{Z}_p$ and $\operatorname{End} E = \mathcal{O}_K$ for some imaginary quadratic field K. Also suppose that $p \nmid \Delta$, where Δ is the discriminant of the cubic polynomial and that p is unramified in $\mathcal{O}_K$. Write $p = \pi\overline{\pi}$. Now the elliptic curve can be reduced modulo p to yield an elliptic curve $\overline{E}/\mathbb{F}_p$. Moreover, $j(\overline{E}) \in \mathbb{F}_p$ satisfies the mod-p reduction of the minimal polynomial of $j(E)$, so $j(\overline{E})$ will necessarily have a linear factor over $\mathbb{F}_p$. The action of π will reduce modulo p to act as the Frobenius endomorphism on $\overline{E}$. In fact, E is the *Serre-Tate canonical lift* of $\overline{E}$.

The *Atkin-Morain CM method* [AM] for generating elliptic curves thus proceeds as follows. First choose a quadratic imaginary field $K/\mathbb{Q}$. Next choose a prime p that is unramified in $\mathcal{O}_K$ and solve for $\pi \in \mathcal{O}_K$ such that $p = \pi\overline{\pi}$. Compute representatives $\mathfrak{a}_i \subseteq \mathcal{O}_K$ for each of the h_K ideal classes and choose an embedding $\iota : K \hookrightarrow \mathbb{C}$. Use (2.6.2) to compute the modular invariants corresponding to the lattices $\iota(\mathfrak{a}_i)$ and form the polynomial (2.6.3). If enough precision was used, the coefficients of the polynomial can be recognized as integers. Reduce the resulting polynomial modulo p and find all roots in $\mathbb{F}_p$. Finally use (2.6.1) to recover an elliptic curves over $\mathbb{F}_p$ with j-invariants equal to these roots. Compute also their quadratic twists $\overline{E}'$. The orders of $\overline{E}(\mathbb{F}_p)$ and $\overline{E}'(\mathbb{F}_p)$ are $\operatorname{Nm}(\pi \mp 1)$, in some order. If one of these norms is almost prime, all that remains to be done is check which one of the twists it is.

Genus 3 Curves

There are two types of genus 3 curves: hyperelliptic and non-hyperelliptic. Hyperelliptic curves are ramified double covers of $\mathbb{P}^1$ and the rest are non-hyperelliptic. Hyperelliptic genus 3 curves over any field of odd or 0 characteristic have an affine model of the form

$$y^2 = x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0\,,$$

which always has a singularity at ∞. This singularity can be resolved to yield an embedding into higher dimensional projective space, and in particular hyperelliptic genus 3 curves are never smooth plane curves. Furthermore, if the characteristic is not equal to 7 an even simpler model

$$y^2 = x\left(x^6 + a_4x^4 + a_2x^2 + a_0\right) \tag{2.6.4}$$

can be obtained. Ramification occurs at the roots of $x(x^6 + a_4x^4 + a_2x^2 + a_0)$ and at ∞, which are called *Weierstrass points* of the curve.

One can show that non-hyperelliptic genus 3 curves are precisely the smooth plane quartic curves, for which the projective embedding is given by the canonical divisor [Har]. This means that the geometry of non-hyperelliptic curves is in some sense simpler than that of hyperelliptic curves and is exploited by Diem to mount an efficient index calculus algorithm against Jacobians of non-hyperelliptic genus 3 curves [Die, Die2, DT, LL], which will be the topic of Chapter 3.

Almost all genus 3 curves are non-hyperelliptic in the following sense.

Theorem 2.6.6. *The moduli space of genus* 3 *curves is* 6*-dimensional and the locus of hyperelliptic genus* 3 *curves is of codimension* 1*.*

Theorem 2.6.7 ([Mil2])**.** *Every* 3*-dimensional principally polarized abelian variety is the Jacobian of a genus* 3 *curve.*

Theorems 2.6.6, 2.6.7, 2.6.1 imply that an arbitrarily chosen 3-dimensional principally polarized abelian variety is "almost certainly" the Jacobian of a non-hyperelliptic curve, so all the attacker needs to do is map the DLP in the moduli space of Jacobians of genus 3 curves to almost any other point and they can expect to obtain a DLP on the Jacobian of a non-hyperelliptic genus 3 curve, which is vulnerable to the fast index calculus attack. These are called *isogeny attacks* and will be discussed further in Chapter 4.

According to Hasse's theorem [Was, Sil1], an elliptic curve $E/\mathbb{F}_p$ satisfies

$$|p + 1 - \#E(\mathbb{F}_p)| \leq 2\sqrt{p}\,,$$

which DLPs in $E(\mathbb{F}_p)$ very good candidates for cryptography. Recall from Section 2.3 that for security against Pollard rho we need p be at least 160 bits. Recall from Example 2.4.3 that this field size is much smaller than what would be needed for a DLP in $\mathbb{F}_p^\times$ due to the lack of an efficient index calculus type algorithm (see [Sem, Sem2] for recent work on elliptic curve index calculus).

The discrete logarithm problem (DLP) in the Jacobian of a genus g curve with $g > 1$ over a finite field $\mathbb{F}_p$ was proposed for use in public key cryptosystems by Koblitz [Kob2]. The potential advantage over elliptic curve cryptosystems comes from the fact that $\#\operatorname{Jac}_C(\mathbb{F}_p) = p^g + O(p^{(2g-1)/2})$. More precisely,

Theorem 2.6.8 (Hasse-Weil)**.** *Let $C/\mathbb{F}_p$ be a hyperelliptic curve of genus g. Then the Jacobian Jac_C is a g-dimensional principally polarized abelian variety with*

$$\left(p^{1/2} - 1\right)^{2g} \leq \#\operatorname{Jac}_C(\mathbb{F}_p) \leq \left(p^{1/2} + 1\right)^{2g}\,.$$

Due to Theorem 2.6.8 the finite field $\mathbb{F}_p$ can be smaller if g is larger, while still yielding a group of the same size with potentially the same security level. In particular, group of $\mathbb{F}_p$-rational points on a Jacobian of a genus 3 hyperelliptic curve is roughly p^3.

Remark 2.6.9. Keep in mind that a genus 3 hyperelliptic curve C itself has only approximately p $\mathbb{F}_p$-rational points.

Generating hyperelliptic genus 2 and genus 3 curves over $\mathbb{F}_p$ with a prescribed number of $\mathbb{F}_p$-points on their Jacobians is significantly more difficult than generating elliptic curves was. In both genus 2 and genus 3 these algorithms were developed and studied by Annegret Weng [Wen, Wen2]. We will only discuss the case of genus 3 and not go into much detail due to the complicated nature of the algorithm. In principle the approach is the same as for elliptic curves:

1) Choose a sextic CM field (recall Definition 2.5.18) K/K_0 where K_0 is a totally real cubic field with class number 1[5] and $O_{K_0} = \mathbb{Z}[w]$, such that $i \in \mathcal{O}_K$[6], and a prime p that is unramified in K[7] and splits as $p = \pi\overline{\pi}$ where $\pi \in \mathcal{O}_K$ is a *p-Weil number*, i.e. each complex embedding has absolute value $p^{1/2}$. Make sure that one of $\mathrm{Nm}\,(\pi \mp 1)$ or $\mathrm{Nm}\,(i\pi \mp 1)$ is divisible by a large prime. The four different Weil numbers $\pm\pi, \pm i\pi$ are associated to four quartic twists of the curve and the above norms give the numbers of $\mathbb{F}_p$-rational points on their Jacobians.

2) Find all *primitive CM types* $\Phi := \{\varphi_1, \varphi_2, \varphi_3\}$ on K [Shim, Wen] and for each of them construct a principal polarization on the period lattice obtained by embedding $\Phi : \mathcal{O}_K \hookrightarrow \mathbb{C}^3$ (see [Wen]). According to Shimura-Taniyama CM theory [Shim] the finite *polarized class group*

$$\frac{\{(\mathfrak{a}, \mu) | \mathfrak{a} \text{ an invertible fractional ideal of } \mathcal{O}_K,\ \mathfrak{a}\overline{\mathfrak{a}} = \mu\mathcal{O}_K,\ \mu \in K_0 \text{ totally positive}\}}{\{(\mu\mathcal{O}_K, \mu\overline{\mu}) | \mu \in K\}}$$

acts freely and transitively on the set of isomorphism classes of isogenous principally polarized abelian varieties of a particular CM type[8]. Using this, it is not hard to find period lattices along with the associated Riemann forms representing isomorphisms classes of every principally polarized abelian variety with CM by $\mathcal{O}_K$.

Remark 2.6.10. Different CM types might embed $\mathcal{O}_K$-ideals into isomorphic lattices and hence yield isomorphic principally polarized abelian varieties, as is explained in [Wen]. In the extreme case where K is a cyclic CM field, every CM type yields an isomorphic set of lattices and hence the same set of principally polarized abelian varieties, so it suffices to consider only one CM type.

[5]This ensures that a principal polarization can be constructed on the period lattice (see [Wen]).
[6]This ensures that we find a hyperelliptic curve (see [Wen] or Theorem 4.4.2).
[7]This ensures that π is an *ordinary* p-Weil number (see the proof of Theorem 4.2.3).
[8]The ideal $\mathfrak{a}$ describes how the lattice changes and μ describes the change in the polarization.

3) Once a complete set of isomorphism classes of principally polarized abelian varieties with CM by $\mathcal{O}_K$ has been found, one can compute analytically the associated theta constants to high precision and consequently recover the Rosenhain invariants of the hyperelliptic curve, as is explained in [Wen].

4) Since $i \in \mathcal{O}_K$, the hyperelliptic curves that will be produced have extra symmetry which significantly simplifies the classical invariant theory of the singular binary octic plane models of the curves [Shio]. Weng [Wen] shows that such curves are classified up to isomorphism by a set of five *absolute Shioda invariants*, which are algebraic numbers over $\mathbb{Q}$ and can be computed to high precision from the Rosenhain invariants.

5) If enough precision was used in computing the absolute Shioda invariants for each CM point the five minimal polynomials (*Shioda class polynomials*) can be recovered. In other words, one can recover the five polynomials, with rational coefficients[9], whose roots are the the five sets of absolute Shioda invariants.

   Remark 2.6.11. Given these five Shioda class polynomials, it is not clear which combinations of their roots yield valid sets of absolute Shioda invariants.

6) Next reduce the Shioda class polynomials modulo the prime p and find all of their roots in $\mathbb{F}_p$. Weng [Wen] explains then how to recover a model over $\mathbb{F}_p$ for the curves corresponding to valid combinations of these $\mathbb{F}_p$-rational absolute Shioda invariants. Remark 2.6.11 still stands, so one must try all combinations to see which ones yield genus 3 hyperelliptic curves over $\mathbb{F}_p$.

7) It remains to form the Jacobians for all hyperelliptic genus 3 curves that have been found. At least one of the curves, or one of its quartic twists, will have Jacobian Jac_C such that $\#\,\mathrm{Jac}_C(\mathbb{F}_p)$ has order divisible by a large prime.

The algorithm has been implemented by Weng and several examples are presented in [Wen].

Genus 3 CM method ([Wen]). *Let K be a sextic CM field with totally real cubic subfield $K_0 := \mathbb{Q}(w)$ such that $i \in \mathcal{O}_K$ and $\mathcal{O}_{K_0} = \mathbb{Z}[w]$. Let p be a prime that is unramified in K and let $\pi \in \mathcal{O}_K$ be a p-Weil number. Suppose K_0 has class number 1 and K has small class number. Then it is possible to find the equation of a hyperelliptic genus 3 curve $C/\mathbb{F}_p$ such that $\mathrm{Jac}_C(\mathbb{F}_p)$ has order $\mathrm{Nm}(\pi \mp 1)$ or $\mathrm{Nm}(i\pi \mp 1)$.*

Remark 2.6.12. Many of the assumptions in the genus 3 CM method can be relaxed. The reader is referred to [Wen] for details.

[9]Shimura [Shim] proves that all complex CM abelian varieties are in fact defined over $\overline{\mathbb{Q}}$, so the Galois group $\mathrm{Gal}(\overline{\mathbb{Q}}/\mathbb{Q})$ acts by permutations in the five sets of absolute Shioda invariants. This proves that the Shioda class polynomials have rational coefficients.

Chapter 3

Index Calculus in Genus 3

3.1 Diem's Index Calculus

There are several variants of Diem's index calculus for low degree plane curves. We restrict to the case of non-hyperelliptic genus 3 curves over finite fields embedded as smooth plane quartics using the canonical embedding [Har]. These are all so-called double large prime index calculus algorithms (recall Section 2.4), where elements of the group are decomposed into sums of factor base elements and at most two non-factor base elements (large primes). The large primes are then eliminated using various methods to produce relations consisting only of factor base elements.

Let C be a non-hyperelliptic curve of genus 3 over a finite field $\mathbb{F}_q$. Using the canonical embedding it can be realized as a smooth plane quartic. Let P_0 be an $\mathbb{F}_q$-rational point on C. We want to find an integer x such that

$$D_2 - 3[P_0] = x \cdot (D_1 - 3[P_0])$$

provided that a solution exists. Here $\deg(D_1) = \deg(D_2) = 3$ and both D_1 and D_2 are sums of three $\mathbb{F}_q$-rational points:

$$D_1 = [P_1^1] + [P_2^1] + [P_3^1]\,, \qquad D_2 = [P_1^2] + [P_2^2] + [P_3^2]\,.$$

For simplicity we assume that both divisors $D_2 - 3[P_0]$ and $D_1 - 3[P_0]$ live in the same subgroup of prime order p.

In general, any $\mathbb{F}_q$-rational divisor can be written in a unique way in the *along P_0 maximally reduced form $D - \ell[P_0]$* (see [Hes]), where $\ell \leq 3$ and in the generic case $\ell = 3$. If D decomposes into a sum of three $\mathbb{F}_q$-rational points, it is called *completely split.* If the divisor is not completely split, the standard strategy is to multiply both sides of the DLP by constants until they become completely split and then proceed as usual. Once the discrete logarithm has been found, these multipliers must be cancelled. We refer the reader to [Die2] for details.

Here is the algorithm as given in [Die2] with one modification. Diem suggests taking the factor base to be slightly larger to ensure that the algorithm terminates successfully. The size used below is an absolute minimum for which we can expect the algorithm to succeed. In practice the size should be slightly bigger.

Choosing the factor base: Choose a set $\mathcal{F} \subseteq C(\mathbb{F}_q)$ with $\lceil ((3/2)\ln q + 4)^{1/2}\, q^{1/2} \rceil$ elements. Include the points $\{P^i_j\} \cup \{P_0\}$ in $\mathcal{F}$. The set $\mathcal{F}$ is called the *factor base*. Let $\mathcal{L} := C(\mathbb{F}_q) \setminus \mathcal{F}$ be the set of *large primes*.

Tree building: We constuct a tree $\mathbf{T}$ as follows.

```
Let $\mathbf{V} = \{*\}$ be the set of vertices of $\mathbf{T}$. Here $*$ denotes a distinguished special vertex. Let
$\mathbf{E} = \emptyset$ be the set of edges.
for all unordered pairs $(F_1, F_2) \in \mathcal{F} \times \mathcal{F}$ such that $F_1 \neq F_2$ do
    Draw a line through the points $F_1$ and $F_2$. This will intersect the curve $C$ at two other
    points $L_1$ and $L_2$.
    if $L_1$ and $L_2$ are not both $\mathbb{F}_q$-rational then
        Move on to the next pair.
    end if
    if $L_1 \in \mathbf{V} \cup \mathcal{F}$ and $L_2 \notin \mathbf{V} \cup \mathcal{F}$ then
        Add a vertex to $\mathbf{T}$ labeled $L_2$.
        If $L_1 \in \mathbf{V}$ draw an edge to $\mathbf{T}$ connecting $L_1$ and $L_2$ and label the edge with the linear
        relation $[F_1] + [F_2] + [L_1] + [L_2] = 0$. If $L_1 \in \mathcal{F}$ draw an edge to $\mathbf{T}$ connecting $*$ and
        $L_2$ and label the edge with the linear relation $[F_1] + [F_2] + [L_1] + [L_2] = 0$.
    end if
    if $\#\mathbf{V} \geq \lceil q^{3/4} \rceil$ then
        break (tree building succeeded)
    end if
end for
if $\#\mathbf{V} < \lceil q^{3/4} \rceil$ then
    Tree building failed. Restart the algorithm with a different set $\mathcal{F}$.
end if
```

Relation search:

```
for all unordered pairs $(F_1, F_2) \in \mathcal{F} \times \mathcal{F}$, $F_1 \neq F_2$, that were not already used in tree
building do
    Draw a line through the points $F_1$ and $F_2$. This will intersect the curve $C$ at two other
    points $L_1$ and $L_2$.
    if $L_1$ and $L_2$ are not both $\mathbb{F}_q$-rational then
        Move on to the next pair.
    end if
    **if** $L_1, L_2 \in \mathbf{V} \cup \mathcal{F}$ **then**
      If one or both of $L_1, L_2$ is in $\mathbf{V}$, use the tree and the linear relations labeling the edges
      to write $[L_1]$ and $[L_2]$ in terms of the factor base. A relation

          $$\sum_{F \in \mathcal{F}} \lambda_F [F] = 0$$

      obtained in this way is called a *full relation.*
    **end if**
    **if** the number of full relations found is $\geq \#\mathcal{F} + 1$ **then**
      **break** (relation search succeeded)
    **end if**
  **end for**
  **if** the number of full relations is $\leq \#\mathcal{F}$ **then**
    Restart the algorithm with a different set $\mathcal{F}$.
  **end if**
```

Linear algebra step:

```
  Construct a sparse matrix $M$ with $\#\mathcal{F}$ columns, each column labeled by a point of $\mathcal{F}$.
  The first row of $M$ is given by the points in the divisor $D_1 - 3[P_0]$.
  The second row of $M$ is given by the points in the divisor $D_2 - 3[P_0]$.
  **for** each full relation **do**
    Add a row to $M$ corresponding to the left-hand side of the full relation.
  **end for**
  Use sparse linear algebra techniques to find a non-zero vector $\overline{v}$ such that $M\overline{v} = \overline{0}$ over the
  ring $\mathbb{Z}/p\mathbb{Z}$ where $p$ is the order of the subgroup of the Jacobian where the DLP was defined.
  Choose $\overline{v}$ to be such that the second component $v_2$ is invertible modulo $p$.
```

Output: $-v_1/v_2$ modulo p.

Theorem 3.1.1 ([Die2]). *Under some justifiable heuristic assumptions and when q is large enough, after a constant number of attempts the algorithm terminates successfully and outputs a number $x \in \mathbb{Z}/p\mathbb{Z}$ such that $D_2 - 3[P_0] = x \cdot (D_1 - 3[P_0])$. The complexity of the algorithm is $\widetilde{O}(q)$.*

3.2 New Variant

In our new improved variant of the algorithm we generate the factor base in a different way. We start with a much smaller initial set for the factor base of size $\widetilde{O}(q^{1/8})$, and then build up the factor base and the graph simultaneously at the beginning. This improves the efficiency of the graph-building, and then after this initial step we build the graph and find full relations simultaneously. As a result, both the overall relation collection time and the linear algebra stage are improved. Our algorithm works as follows.

Input:

1) The Jacobian of a smooth plane quartic C over a finite field $\mathbb{F}_q$;
2) An $\mathbb{F}_q$-rational point P_0 on the curve;
3) A discrete logarithm problem on the Jacobian

$$D_2 - 3[P_0] = x \cdot (D_1 - 3[P_0])$$

   where $\deg(D_1) = \deg(D_2) = 3$ and both D_1 and D_2 are sums of three $\mathbb{F}_q$-rational points:

$$D_1 = [P_1^1] + [P_2^1] + [P_3^1]\,, \qquad D_2 = [P_1^2] + [P_2^2] + [P_3^2]\,;$$

4) The size p of a prime order subgroup[1] containing $D_2 - 3[P_0]$ and $D_1 - 3[P_0]$.

Initialization: Let λ be a positive real number satisfying

$$\lambda \exp\left(4\lambda^8\right) = q^{1/8}\,.$$

Choose a set $\mathcal{RP}$ of $\lceil 4\lambda\, q^{1/8} \rceil$ $\mathbb{F}_q$-rational points on the curve. Let $\mathcal{F}$ be another set of points, the *factor base*, and for now let

$$\mathcal{F} := \mathcal{RP} \cup \{P_j^i\} \cup \{P_0\}\,.$$

Construction of the base vertices: We construct a graph $\mathbf{G}$ as follows.

```
Let $\mathbf{V} := \{*\}$ be the set of vertices of the graph $\mathbf{G}$. Here $*$ denotes a distinguished *special vertex*. Let $\mathbf{E} := \emptyset$ be the set of edges.
**for** all unordered pairs $(F_1, F_2) \in \mathcal{RP} \times \mathcal{RP}$ such that $F_1 \neq F_2$ **do**
    Draw a line through the points $F_1$ and $F_2$. This will intersect the curve $C$ at two other points $F$ and $L$.
    **if** $F$ and $L$ are $\mathbb{F}_q$-rational, $L \notin \mathbf{V} \cup \mathcal{F}$ and $F \notin \mathbf{V}$ **then**
        Let $\mathcal{F} := \{F\} \cup \mathcal{F}$ and add a vertex to $\mathbf{V}$ labeled $L$.
        Add an edge to $\mathbf{E}$ connecting $*$ and $L$ and label the edge with the linear relation $[F_1] + [F_2] + [F] + [L] = 0$. The vertices $L$ constructed here we call *base vertices*.
    **end if**
**end for**
```

[1]It is not strictly speaking necessary for the subgroup to have prime order, but this will simplify the linear algebra step and guarantee that the DLP has a solution.

Let $\mathbf{B}$ denote the set of all base vertices. Note that at this point $\mathbf{V} = \mathbf{B} \cup \{*\}$.

Construction of the triangle relations:

```
for all unordered pairs $(B_1, B_2) \in \mathbf{B} \times \mathbf{B}$ such that $B_1 \neq B_2$ do
    Draw a line through the points $B_1$ and $B_2$. This will intersect the curve $C$ at two other
    points $F$ and $L$.
    if $F$ and $L$ are $\mathbb{F}_q$-rational, $L \notin \mathbf{V} \cup \mathcal{F}$ and $F \notin \mathbf{V}$ then
        Let $\mathcal{F} := \{F\} \cup \mathcal{F}$ and add a vertex to $\mathbf{V}$ labeled $L$.
        Draw a triangle in the graph with corners $B_1$, $B_2$ and $L$. Label the triangle with the
        linear relation $[B_1] + [B_2] + [L] + [F] = 0$. The vertices $L$ constructed here we call *top
        triangle vertices*.
    end if
end for
```

Graph building and relation search (RS):

```
for all unordered pairs $(F_1, F_2) \in \mathcal{F} \times \mathcal{F}$ such that $F_1 \neq F_2$ do
    Draw a line through the points $F_1$ and $F_2$. This will intersect the curve $C$ at two other
    points $L_1$ and $L_2$.
    if $L_1$ and $L_2$ are not both $\mathbb{F}_q$-rational then
        Move on to the next pair.
    end if
    if $L_1 \in \mathbf{V} \setminus \mathbf{B}$ and $L_2 \notin \mathbf{V} \cup \mathcal{F}$ then
        Add a vertex to $\mathbf{V}$ labeled $L_2$.
        Add an edge to $\mathbf{E}$ connecting $L_1$ and $L_2$ and label the edge with the linear relation
        $[F_1] + [F_2] + [L_1] + [L_2] = 0$.
    else if $L_1, L_2 \in (\mathbf{V} \setminus \mathbf{B}) \cup \mathcal{F}$ then
        In case $L_1, L_2 \in \mathbf{V} \setminus \mathbf{B}$, use the graph and the linear relations labeling the edges to
        write $[L_1]$ and $[L_2]$ in terms of the factor base to obtain a relation

            $$\lambda_1[L'_1] + \lambda_2[L'_2] + \sum_{F \in \mathcal{F}} \lambda_F[F] = 0\,,$$

        where $L'_1$ and $L'_2$ are top triangle vertices, $\lambda_i$ are $\pm 1$ and $\lambda_F$ are integers. Then use
        the triangle relations to substitute $L'_1$ and $L'_2$ with elements of $\mathcal{F}$. In cases where
        neither one or exactly one of $L_i$ is in $\mathbf{V} \setminus \mathbf{B}$ we need to perform the above substitution
        process only to the one that is in $\mathbf{V} \setminus \mathbf{B}$. In any case we obtain a relation involving
        only elements of $\mathcal{F}$, which we record. We call it a *full relation*.
    end if
    if the number of full relations found is $\geq \#\mathcal{F} + 1$ then
        break (relation search succeeded)
    end if
end for
if the number of full relations is $\leq \#\mathcal{F}$ then
    Restart the algorithm.
end if
```

In practice we do not restart the algorithm but instead re-run the **for**-loop. Almost certainly it will **break** very soon after starting and only a few if any duplicate full relations are produced.

Linear algebra step:

```
Construct a sparse matrix $M$ with $\#\mathcal{F}$ columns, each column labeled by a point of $\mathcal{F}$.
The first row of $M$ is given by the points in the divisor $D_1 - 3[P_0]$.
The second row of $M$ is given by the points in the divisor $D_2 - 3[P_0]$.
for each full relation do
    Add a row to $M$ corresponding to the left-hand side of the full relation.
end for
Use sparse linear algebra techniques to find a non-zero vector $\overline{v}$ such that $M\overline{v} = \overline{0}$ over the ring $\mathbb{Z}/p\mathbb{Z}$ where $p$ is the order of the subgroup of the Jacobian where the DLP was defined.
Choose $\overline{v}$ to be such that the second component $v_2$ is invertible modulo $p$.
```

Output: $-v_1/v_2$ modulo p.

Theorem 3.2.1. *(Heuristic) Under some justifiable heuristic assumptions and if q is large enough, after a constant number of attempts the algorithm terminates and outputs a number $x \in \mathbb{Z}/p\mathbb{Z}$ such that $D_2 - 3[P_0] = x \cdot (D_1 - 3[P_0])$. The size of the factor base will be approximately $4\lambda^4 q^{1/2}$ and the size of the graph will be approximately $4\lambda^2 q^{3/4}$.*

Proof. Correctness is easy; see for example [Die2]. It remains to prove that the size of the factor base is sufficiently large for the algorithm to terminate. The strategy is the following. Let N be the size of the factor base at the beginning, essentially $N = \#\mathcal{RP}$. First we compute the number of factor base elements and graph vertices produced when constructing the base vertices and the triangle relations, and express these in terms of N. Next we compute the expected number of full relations produced in the graph building step when allowing the graph to grow until it has $N_{\max}$ vertices. Since we know that the number of full relations needed is $\#\mathcal{F} + 1$, we can find an expression for $N_{\max}$ in terms of N. Finally we compute the number of factor base pairs needed to grow the graph to size $N_{\max}$. We set this number equal to the number of factor base pairs available, which yields an equation for N.

Due to the roundabout way of computing the numbers N and $N_{\max}$, we need to have an idea of their sizes beforehand, so that the most significant terms can be computed. For this purpose, we assume $N = \widetilde{O}(q^{1/8})$, so $\#\mathcal{F} = \widetilde{O}(q^{1/2})$, and $N_{\max} = \widetilde{O}(q^{3/4})$, which are in line with Diem's choices in [Die2].

By [DT, Prop. 14], a line through two $\mathbb{F}_q$-rational points on the curve intersects the curve at two other $\mathbb{F}_q$-rational points with probability $1/2 + O(q^{-1/2})$.

Suppose $\mathcal{RP}$ is a set of $N = \widetilde{O}(q^{1/8})$ random $\mathbb{F}_q$-rational points on the curve. Construction of the base vertices $\mathbf{B}$ produces

$$\left(\frac{1}{2} + O(q^{-1/2})\right)\binom{N}{2} = \frac{N^2}{4} + \widetilde{O}(q^{1/8})$$

base vertices and equally many factor base elements.

Construction of the triangle relations from the set $\mathbf{B}$ produces

$$\left(\frac{1}{2} + O(q^{-1/2})\right)\binom{\#\mathbf{B}}{2} = 4\left(\frac{N}{4}\right)^4 + \widetilde{O}(q^{3/8})$$

triangles and equally many factor base elements. At this point we expect to have

$$\#\mathcal{F} = 4\left(\frac{N}{4}\right)^4 + \widetilde{O}(q^{3/8}) = 4\left(\frac{N}{4}\right)^4\left(1 + \widetilde{O}(q^{-1/8})\right).$$

We will need to choose N so that in the graph building step we expect to find $\#\mathcal{F}+1$ full relations. To this end, suppose that when the algorithm terminates successfully the number of vertices in the graph is $N_{\max} = \widetilde{O}(q^{3/4})$. When the graph building starts we already have approximately $4(N/4)^4$ vertices in the graph, consisting of the base vertices $\mathbf{B}$ and the top triangle vertices. If the size of the graph at a particular moment is x, the probability of adding a new vertex and edge to the graph with a particular choice of a pair $(F_1, F_2) \in \mathcal{F}\times\mathcal{F}$ is

$$\left(1 + O(q^{-1/2})\right)\frac{x}{q}\left(1 - \frac{x}{q}\right).$$

Hence, to add one new vertex and edge we need to try approximately

$$\left(1 + O(q^{-1/2})\right)\frac{1}{(x/q)(1 - x/q)}$$

pairs. For each pair we try, the probability of finding a full relation through the process described in the algorithm is

$$\left(\frac{1}{2} + O(q^{-1/2})\right)\left(\frac{x}{q}\right)^2$$

so when the size of the graph is increased by one we expect to have found approximately

$$\left(\frac{1}{2} + O(q^{-1/2})\right)\frac{x/q}{1 - x/q}$$

more full relations.

Once the triangle building step has been completed, the size of the graph is roughly equal to the number of triangles, which again is roughly equal to the size of the factor base. We denote this by

$$N_0 = 4\left(\frac{N}{4}\right)^4 + \widetilde{O}(q^{3/8})\,.$$

Note that once the triangles have been constructed, the factor base does not change anymore and only the graph is built using the factor base. The total number of full relations produced in the entire graph building step, when the size of the graph is built from N_0 to $N_{\max} = \widetilde{O}(q^{3/4})$, is

$$\left(\frac{1}{2} + O(q^{-1/2})\right) \sum_{k=N_0}^{N_{\max}-1} \frac{k/q}{1-k/q} = \left(\frac{1}{2} + O(q^{-1/2})\right)\left(\int_{N_0}^{N_{\max}} \frac{x/q}{1-x/q}\,dx + E\right)$$

$$= \left(\frac{q}{2} + O(q^{1/2})\right)\left[-\frac{N_{\max}}{q} - \ln\left(1 - \frac{N_{\max}}{q}\right) + \frac{N_0}{q} + \ln\left(1 - \frac{N_0}{q}\right) + \frac{E}{q}\right], \qquad (3.2.1)$$

where the error term E is

$$E = \sum_{k=N_0}^{N_{\max}-1}\left(\frac{k/q}{1-k/q} - \int_k^{k+1} \frac{x/q}{1-x/q}\,dx\right).$$

Since the function $(k/q)/(1-k/q)$ is increasing, E satisfies

$$\sum_{k=N_0}^{N_{\max}-1}\left(\frac{k/q}{1-k/q} - \frac{(k+1)/q}{1-(k+1)/q}\right) = \frac{N_0/q}{1-N_0/q} - \frac{N_{\max}/q}{1-N_{\max}/q} \le E \le 0\,.$$

Hence $E \in \widetilde{O}(q^{-1/4})$ and $E/q \in \widetilde{O}(q^{-5/4})$. If we expand out the first two terms of the logarithms in (3.2.1), we find that the total number of full relations produced in the entire graph building step is

$$\left(\frac{q}{2} + O(q^{1/2})\right)\left[\frac{1}{2}\left(\frac{N_{\max}}{q}\right)^2 + \widetilde{O}(q^{-3/4})\right] = \frac{q}{4}\left(\frac{N_{\max}}{q}\right)^2 + \widetilde{O}(q^{1/4})\,.$$

We want this to equal roughly the size of the factor base (or maybe slightly more in practice to ensure that the algorithm terminates successfully) so we must have

$$\frac{q}{4}\left(\frac{N_{\max}}{q}\right)^2 = 4\left(\frac{N}{4}\right)^4\left(1 + \widetilde{O}(q^{-1/8})\right).$$

From this we can solve

$$\frac{N_{\max}}{4q} = \left(\frac{N}{4q^{1/4}}\right)^2\left(1 + \widetilde{O}(q^{-1/8})\right) = \left(\frac{N}{4q^{1/4}}\right)^2 + \widetilde{O}(q^{-3/8})\,. \qquad (3.2.2)$$

With (3.2.2) we are ready to compute the value of N. The number of unordered pairs of factor base elements needed to build the graph from size N_0 to size $N_{\max} = \widetilde{O}(q^{3/4})$ should equal the number of pairs available, i.e. $\binom{\#\mathcal{F}}{2} = 8(N/4)^8 + \widetilde{O}(q^{7/8})$. Therefore,

$$\begin{aligned} 8\left(\frac{N}{4}\right)^8 + \widetilde{O}(q^{7/8}) &= \left(1 + O(q^{-1/2})\right) \sum_{k=N_0}^{N_{\max}-1} \frac{1}{(k/q)(1-k/q)} \\ &= \left(1 + O(q^{-1/2})\right) \left(\int_{N_0}^{N_{\max}} \frac{1}{(x/q)(1-x/q)}\, dx + E \right). \end{aligned} \tag{3.2.3}$$

The error term E is

$$E = \sum_{k=N_0}^{N_{\max}-1} \left(\frac{1}{(k/q)(1-k/q)} - \int_k^{k+1} \frac{1}{(x/q)(1-x/q)}\, dx \right).$$

Since the function $1/((k/q)(1-k/q))$ is decreasing, E satisfies

$$0 \le E \le \frac{1}{(N_0/q)(1-N_0/q)} - \frac{1}{(N_{\max}/q)(1-N_{\max}/q)}$$

so $E \in \widetilde{O}(q^{1/2})$. Now (3.2.3) becomes

$$\begin{aligned} &\left(q + O(q^{1/2})\right) \left[\ln\left(\frac{N_{\max}}{4q}\right) - \ln\left(1 - \frac{N_{\max}}{q}\right) - \ln\left(\frac{N_0}{4q}\right) + \ln\left(1 - \frac{N_0}{q}\right) + \frac{E}{q} \right] \\ &= \left(q + O(q^{1/2})\right) \left[\ln\left(\frac{N_{\max}}{4q}\right) - \ln\left(\frac{N_0}{4q}\right) + \widetilde{O}(q^{-1/4}) \right] \\ &= -2q \ln\left(\frac{N}{4q^{1/4}}\right) + \widetilde{O}(q^{7/8}) \end{aligned}$$

where we have used (3.2.2). Hence we obtain the equation

$$4\left(\frac{N}{4q^{1/8}}\right)^8 + \widetilde{O}(q^{-1/8}) = -\ln\left(\frac{N}{4q^{1/4}}\right).$$

Denoting

$$\lambda = \frac{N}{4q^{1/8}}$$

this becomes

$$\lambda \exp\left(4\lambda^8\right) = q^{1/8} + \widetilde{O}(1) \approx q^{1/8},$$

where the approximation makes sense when q is large enough. In practice the error term is small, even for small field sizes. The function $\lambda \exp(4\lambda^8) - q^{1/8}$ is monotonically increasing and has precisely one positive real root. This is the equation stated in the initialization step of the algorithm, so if we take $N = 4\lambda\, q^{1/8}$ the algorithm can be expected to terminate successfully.

If q is large, then the size of the factor base will be

$$\#\mathcal{F} = 4\lambda^4\, q^{1/2} + \widetilde{O}(q^{3/8}) \approx 4\lambda^4\, q^{1/2}$$

and the size of the graph when the algorithm terminates will be

$$N_{\max} = 4\lambda^2\, q^{3/4} + \widetilde{O}(q^{5/8}) \approx 4\lambda^2\, q^{3/4}\,.$$

$\square$

For field sizes of most practical interest (perhaps between 60 and 120 bits) our algorithm has not much worse storage requirements than Diem's algorithm but the factor base remains smaller. In practice the factor base can be taken to be even slightly smaller than $4\lambda^4\, q^{1/2}$ if we run the graph building step twice in a row as was explained earlier. We will look at some precise numbers in the next section.

Specifics of Implementation in Characteristic 2

In our experiments we made an artificial restriction to the case of binary fields in order to be able to count the number of points on the Jacobian easily. In this case it is easy to perform the geometric step of the algorithm where a line is drawn through two points on the curve and the intersection divisor is computed. Indeed, we did this by first constructing an equation for the line, then solving for one of the variables in terms of the other ones and substituting this into the equation of the curve to obtain a quartic polynomial in one variable. It is a simple computation to divide this polynomial by the two linear factors corresponding to the two points we started with. Finally the quadratic equation can be transformed into an Artin-Schreier equation by a linear change of variables and the solutions can be immediately written down using Chen's formulas [Che]. The complexity is logarithmic in q.

For odd characteristic fields one has to compute a square-root in $\mathbb{F}_q$ using the Tonelli-Shanks algorithm to solve the quadratic polynomial, the complexity of which is logarithmic in q.

3.3 Complexity for Realistic Field Sizes

Relation Collection Time and Total Memory

In our naive implementation the number of field multiplications (in the binary field case) needed to process each pair of factor base elements as explained above was

$$M_{\text{pair}} := 7\log_2 q + 13\,.$$

This count is an actual count of how many field multiplications are used to process each pair of points in our code, but it is also a theoretical count of the number of field operations required to pass a line through two points and intersect it with the curve. There are field inversions involved, which roughly explains the $\log q$ factors. This number could likely be improved in a more robust implementation. The total number of field multiplications needed was therefore approximately

$$M_{\text{Total}} \approx M_{\text{pair}} \cdot \binom{\#\mathcal{F}}{2} = (7\log_2 q + 13) \cdot 8\lambda^8 \, q \,.$$

Remark 3.3.1. Locally the numbers λ behave roughly logarithmically as a function of q. If we focus on the range $q \in [2^{70}, 2^{120}]$ we can for instance approximate

$$\lambda(q) \approx C \log_2^{\alpha} q \,, \quad \text{where} \quad C = 0.62054 \,, \quad \alpha = 0.12431 \,.$$

Then

$$M_{\text{Total}} \approx 0.17589 \cdot (7\log_2 q + 13) \cdot \log_2^{0.99448}(q) \cdot q \approx 1.23123 \cdot \log_2^{2}(q) \cdot q \,.$$

The memory consumption was roughly 370 bytes per graph vertex but our implementation was not optimized towards saving memory so this number can probably be reduced by a significant factor. Moreover, every vertex data must contain the coordinates of some points on the curve or other vertex identifiers, the size of which grows logarithmically in q. Hence the size of a vertex will grow logarithmically in q, but this dependence is weak and for practical field sizes it is not a significant factor. To take this into account, we assume the size of a vertex data is $\lceil (\log_2 q)/64 \rceil \cdot 370$ bytes. The results are shown in Table 3.1.

According to these estimates one should use a field of size at least 115 bits to get a security level of 128 bits and a field of size at least 240 bits to get a security level of 256 bits. Of course this is only the relation collection step and is not taking into account the linear algebra step or even more importantly the massive memory consuption. Later we will see that there is a time-memory trade-off which can be used to reduce the memory cost significantly without affecting the computational complexity much, and that parallelization can be used to even further reduce both time and memory requirements (per computer).

Experimental Results for Small Examples

We ran small experiments and got results corresponding to the theoretical results above. We chose $\mathcal{RP}$ to be slightly bigger than was strictly speaking needed to ensure that the algorithm finishes successfully on the first attempt. More precisely, we chose it so that about 95% of the factor base pairs are used when the algorithm finishes. The results are shown in Table 3.2. We conclude that the experimental results correspond closely to the theoretical results, even for such small field sizes.

Table 3.1: Results for field sizes of practical interest

Field size	λ	$\log_q \#\mathcal{F}$	$\log_q N_{\max}$	$\log_q M_{\text{Total}}$	$\log_2 M_{\text{Total}}$	Memory
2^{30}	0.94987	0.55677	0.81172	1.34024	40.20729	7.4 GB
2^{40}	0.98285	0.54750	0.79875	1.27488	50.99510	1430 GB
2^{50}	1.00974	0.54112	0.79056	1.23231	61.61568	267 TB
2^{60}	1.03251	0.53641	0.78487	1.20212	72.12744	50490 TB
2^{70}	1.05230	0.53277	0.78067	1.17947	82.56275	$1.90 \cdot 10^7$ TB
2^{80}	1.06983	0.52987	0.77743	1.16177	92.94144	$3.56 \cdot 10^9$ TB
2^{90}	1.08559	0.52749	0.77486	1.14752	103.27649	$6.62 \cdot 10^{11}$ TB
2^{100}	1.09992	0.52550	0.77275	1.13577	113.57690	$1.2 \cdot 10^{14}$ TB
2^{110}	1.11306	0.52380	0.77099	1.12590	123.84916	$2.28 \cdot 10^{16}$ TB
2^{115}	1.11926	0.52304	0.77022	1.12153	128.97628	$3.10 \cdot 10^{17}$ TB
2^{120}	1.12522	0.52234	0.76950	1.11748	134.09806	$4.22 \cdot 10^{18}$ TB
2^{140}	1.14712	0.51994	0.76711	1.10386	154.53975	$2.16 \cdot 10^{23}$ TB
2^{160}	1.16646	0.51805	0.76528	1.09327	174.92298	$7.29 \cdot 10^{27}$ TB
2^{180}	1.18380	0.51652	0.76382	1.08479	195.26141	$8.43 \cdot 10^{31}$ TB
2^{200}	1.19954	0.51525	0.76262	1.07782	215.56441	$1.10 \cdot 10^{37}$ TB
2^{220}	1.21397	0.51418	0.76163	1.07199	235.83869	$3.71 \cdot 10^{41}$ TB
2^{240}	1.22729	0.51326	0.76080	1.06704	256.08921	$1.24 \cdot 10^{46}$ TB

Table 3.2: Experimental results. (FBPU := percent of factor base pairs used; th/pr denote the value suggested by theory/value observed in practice)

Field size	$\log_q \#\mathcal{F}$ (th)	$\log_q \#\mathcal{F}$ (pr)	FBPU	$\log_q M_{\text{Total}}$ (th)	$\log_q M_{\text{Total}}$ (pr)
2^{17}	0.57846	0.58034	96.4	1.51247	1.50780
2^{19}	0.57387	0.57692	95.5	1.47352	1.46740
2^{21}	0.56683	0.57223	95.3	1.44080	1.43510
2^{23}	0.56637	0.56819	95.4	1.41287	1.40769
2^{25}	0.56325	0.56468	96.0	1.38869	1.38418
2^{27}	0.56046	0.56158	96.4	1.36752	1.36351

Linear Algebra

The complexity of sparse linear algebra algorithms is $O\left(w \cdot (\#\mathcal{F})^2\right)$ where w denotes the average row weight. The row weight depends logarithmically on the field size. The expected average depth of a tree on a given number of vertices can be estimated using the method in [GTTD] but we need to take into account that the size of the graph is changing while we are looking for full relations.

Lemma 3.3.2. *The average row weight of the matrix is $w \le 18 - 8\ln\lambda + \ln q$.*

Proof. In [GTTD] it is established that for a tree containing k vertices the expected average depth can be approximated from above by the function $1 + \ln k$. Our graph consists of trees built on top of each of the top triangle vertices. When our graph has a total of k vertices, each one of these $4\lambda^4 q^{1/2} + \widetilde{O}(q^{3/8})$ trees has on average $\left(1 + \widetilde{O}(q^{-1/8})\right) \cdot k/(4\lambda^4 q^{1/2})$ vertices. So if we choose a tree at random and a point in it at random, the expected depth (measured from the root of that tree) is at most

$$1 + \ln\left(\frac{k}{4\lambda^4 q^{1/2}}\right) + \widetilde{O}(q^{-1/8}) .$$

We find full relations while building the graph so the sizes of the trees change. Recall that right after the triangles are constructed, the size of the graph is

$$N_0 = 4\lambda^4 q^{1/2} + \widetilde{O}(q^{3/8}) .$$

When a full relation is produced and the graph tracing step performed, the number of factor base elements we end up with on average is at most[2]

$$\begin{aligned}
&\frac{2^2}{\#\mathcal{F}} \sum_{k=N_0}^{N_{\max}-1} \left(1 + \ln\left(\frac{k}{4\lambda^4 q^{1/2}}\right) + \widetilde{O}(q^{-1/8})\right) \cdot \left(\frac{1}{2} + O(q^{-1/2})\right) \frac{k/q}{1-k/q} \\
&= \frac{q^{1/2}}{2\lambda^4}\left(1 + \widetilde{O}(q^{-1/8})\right)\left[\int_{N_0/q}^{N_{\max}/q} \left(1 + \ln\left(\frac{t\,q^{1/2}}{4\lambda^4}\right) + \widetilde{O}(q^{-1/8})\right) \cdot \frac{t}{1-t}\,dt + \frac{E}{q}\right] \\
&= \frac{q^{1/2}}{2\lambda^4}\left(1 + \widetilde{O}(q^{-1/8})\right)\left[\left(1 + \widetilde{O}(q^{-1/4})\right)\int_{N_0/q}^{N_{\max}/q} t\left(1 + \ln\left(\frac{t\,q^{1/2}}{4\lambda^4}\right) + \widetilde{O}(q^{-1/8})\right) dt + \frac{E}{q}\right] \\
&= 4\left(1 + \widetilde{O}(q^{-1/8})\right)\left[\left(\frac{1}{2} - 2\ln\lambda + \frac{1}{4}\ln q + \widetilde{O}(q^{-1/8})\right) + \widetilde{O}(q^{-1/2}) + \frac{E}{8\lambda^4 q^{1/2}}\right] .
\end{aligned}$$

Here the error term E is

$$\begin{aligned}
E = \sum_{k=N_0}^{N_{\max}-1} &\left[\left(1 + \ln\left(\frac{k}{4\lambda^4 q^{1/2}}\right) + \widetilde{O}(q^{-1/8})\right)\frac{k/q}{1-k/q}\right. \\
&\left.- \int_k^{k+1} \left(1 + \ln\left(\frac{x}{4\lambda^4 q^{1/2}}\right) + \widetilde{O}(q^{-1/8})\right)\frac{x/q}{1-x/q}\,dx\right]
\end{aligned}$$

and it satisfies the bounds

$$\left(1 + \ln\left(\frac{N_0}{4\lambda^4 q^{1/2}}\right) + \widetilde{O}(q^{-1/8})\right)\frac{N_0/q}{1-N_0/q}$$

[2]One factor of 2 is for the two factor base elements that are produced at every step in the graph and the other factor of 2 from tracing back two vertices to their respective roots. Here we are only concerned with the factor base elements that result from moving in the graph. The others are taken into account below.

$$-\left(1+\ln\left(\frac{N_{\max}}{4\lambda^4\,q^{1/2}}\right)+\widetilde{O}(q^{-1/8})\right)\frac{N_{\max}/q}{1-N_{\max}/q}\le E\le 0\,.$$

Hence $E\in\widetilde{O}(q^{-1/4})$ and $E/(8\lambda^4\,q^{1/2})\in\widetilde{O}(q^{-3/4})$. Finally, with this we find that the average number of factor base elements appearing is

$$2-8\ln\lambda+\ln q+\widetilde{O}(q^{-1/8})+\widetilde{O}(q^{-3/4})=2-8\ln\lambda+\ln q+\widetilde{O}(q^{-1/8})\,.$$

In addition to the contribution coming from moving in the graph, we have 2 initial factor base elements, 2 coming from using the triangle relations and 12 from the base vertices. The expected average row weight should therefore satisfy $w\le 18-8\ln\lambda+\ln q$. $\square$

Note that in practice q is large so $q^{-1/8}$ is small, and we let $w_{\text{est}} := 18-8\ln\lambda+\ln q$. Results for field sizes of practical interest are shown in Table 3.3. The complexity of the linear algebra is in all cases slightly better than the complexity of relation search so there might be some room for optimization by choosing the factor base to be slightly larger, which makes relation collection faster and linear algebra slower.

Table 3.3: Linear algebra results for field sizes of practical interest

Field size	$\log_2 w_{\text{est}}$	$\log_2 \#\mathcal{F}$	Lin. alg.	M_{Total}
2^{30}	5.29300	16.70320	$\approx 2^{39}$	$\approx 2^{40}$
2^{40}	5.51930	21.90017	$\approx 2^{49}$	$\approx 2^{51}$
2^{50}	5.71644	27.05593	$\approx 2^{60}$	$\approx 2^{62}$
2^{60}	5.89076	32.18461	$\approx 2^{70}$	$\approx 2^{72}$
2^{70}	6.04685	37.29417	$\approx 2^{81}$	$\approx 2^{83}$
2^{80}	6.18808	42.38952	$\approx 2^{91}$	$\approx 2^{93}$
2^{90}	6.31698	47.47391	$\approx 2^{101}$	$\approx 2^{103}$
2^{100}	6.43551	52.54957	$\approx 2^{112}$	$\approx 2^{114}$
2^{110}	6.54518	57.61814	$\approx 2^{122}$	$\approx 2^{124}$
2^{115}	6.59709	60.15016	$\approx 2^{127}$	$\approx 2^{129}$
2^{120}	6.64723	62.68083	$\approx 2^{132}$	$\approx 2^{134}$
2^{140}	6.83216	72.79205	$\approx 2^{152}$	$\approx 2^{155}$
2^{160}	6.99630	82.88852	$\approx 2^{173}$	$\approx 2^{175}$
2^{180}	7.14381	92.97370	$\approx 2^{193}$	$\approx 2^{195}$
2^{200}	7.27774	103.04993	$\approx 2^{213}$	$\approx 2^{216}$
2^{220}	7.40038	113.11892	$\approx 2^{234}$	$\approx 2^{236}$
2^{240}	7.51347	123.18192	$\approx 2^{254}$	$\approx 2^{256}$

3.4 Time-Memory Trade-offs

Our variant presented in Section 3.2 improves on the computational complexity of Diem's algorithm by building a larger graph. Since the memory costs of algorithms of this type are enormous, one should ask how the computational complexity behaves when the size of the graph is limited to something between that of Diem's algorithm ($q^{3/4}$ vertices) and $N_{\max}$ given in Theorem 3.2.1.

Let χ be a fixed positive real number. Suppose in our algorithm we let our graph grow to size $\chi\, q^{3/4}$ and after that only search for full relations. More precisely, we start by choosing a set $\mathcal{RP}_\chi$ of $4\eta_\chi\, q^{1/8}$ random points on the curve where η_χ is a real number depending on χ and use these to construct a factor base $\mathcal{F}_\chi$ of size $4\eta_\chi^4\, q^{1/2}$ just as in our original algorithm. We expect η_χ to be somewhere between 1 and 10 for practical field sizes. In the graph building/relation searching step we stop adding new vertices to the graph once it has reached size $\chi\, q^{3/4}$ and after that only search for full relations. To simplify the notation we denote η_χ just by η.

Theorem 3.4.1. *(Heuristic) If q is large enough and we take η to be a root of*

$$2\eta^2 \exp\left(4\eta^8 - 4\eta^4/\chi^2 + 1/4\right) = \chi^{1/2}\, q^{1/8} \tag{3.4.1}$$

we can expect the algorithm to terminate successfully. The average row weight of the matrix will be approximately

$$w_{est}^{\chi} := 20 - \frac{\chi^2}{8\eta^4} - 4\ln(4\eta^4) + \ln q + 4\ln\chi\,. \tag{3.4.2}$$

Proof. This is very similar to the proof of Theorem 3.2.1. We start with a set $\mathcal{RP}_\chi$ of $4\eta\, q^{1/8} = \widetilde{O}(q^{1/8})$ random points. The size of the factor base and the number of vertices in the graph after the triangles have been constructed are

$$\#\mathcal{F}_\chi = 4\eta^4\, q^{1/2} + \widetilde{O}(q^{3/8})\,, \qquad N_0 = 4\eta^4\, q^{1/2} + \widetilde{O}(q^{3/8})\,.$$

We build the graph until it has size $\chi\, q^{3/4}$. This produces

$$\left(\frac{1}{2} + O(q^{-1/2})\right) \sum_{k=N_0}^{\chi\, q^{3/4}-1} \frac{k/q}{1-k/q} = \frac{\chi^2\, q^{1/2}}{4} + \widetilde{O}(q^{1/4})$$

full relations (see the proof of Theorem 3.2.1). The total number of full relations needed is $\#\mathcal{F}_\chi + 1$ so we are lacking

$$4\eta^4\, q^{1/2} - \frac{\chi^2\, q^{1/2}}{4} + \widetilde{O}(q^{3/8}) = \left(4\eta^4 - \frac{\chi^2}{4}\right) q^{1/2} + \widetilde{O}(q^{3/8})$$

of them. We also need to know how many pairs of factor base elements were used in this. Just like in the proof of Theorem 3.2.1 we find that this number is

$$\left(1 + O(q^{-1/2})\right) \sum_{k=N_0}^{\chi\, q^{3/4}-1} \frac{1}{(k/q)(1-k/q)} = q \ln\left(\frac{\chi\, q^{1/4}}{4\eta^4}\right) + \widetilde{O}(q^{7/8})\,.$$

Now once the graph has size $\chi\, q^{3/4}$ we need to find the rest of the full relations. For each pair of factor base elements we try, the probability of finding a full relation is now approximately $\chi^2/(2q^{1/2}) + \widetilde{O}(q^{-1})$ (see the proof of Theorem 3.2.1). Thus we need to try

$$\frac{2q^{1/2}}{\chi^2}\left(1+O(q^{-1/2})\right)\cdot\left[\left(4\eta^4-\frac{\chi^2}{4}\right)q^{1/2}+\widetilde{O}(q^{3/8})\right]=\left(\frac{8\eta^4}{\chi^2}-\frac{1}{2}\right)q+\widetilde{O}(q^{7/8})$$

more factor base pairs. We want to end up using precisely all of the factor base pairs, of which there are $\binom{\#\mathcal{F}_\chi}{2} = 8\eta^8\, q + \widetilde{O}(q^{7/8})$. We get the equation

$$8\eta^8\, q+\widetilde{O}(q^{7/8}) = q\ln\left(\frac{\chi\, q^{1/4}}{4\eta^4}\right)+\left(\frac{8\eta^4}{\chi^2}-\frac{1}{2}\right)q+\widetilde{O}(q^{7/8})\,.$$

Exponentiating this yields

$$2\eta^2\,\exp(4\eta^8-4\eta^4/\chi^2+1/4)=\chi^{1/2}\,q^{1/8}+\widetilde{O}(1)\,.$$

When q is big we can approximate this with

$$2\eta^2\,\exp(4\eta^8-4\eta^4/\chi^2+1/4)=\chi^{1/2}\,q^{1/8}\,.$$

As in the proof of Theorem 3.3.2 we find that the average row weight is approximated from above by

$$\begin{aligned}
16+\frac{2^2}{\#\mathcal{F}_\chi}&\left[\sum_{k=N_0}^{\chi\, q^{3/4}-1}\left(1+\ln\left(\frac{k}{4\eta^4\, q^{1/2}}\right)+\widetilde{O}(q^{-1/8})\right)\cdot\left(\frac{1}{2}+O(q^{-1/2})\right)\frac{k/q}{1-k/q}\right.\\
&\left.+\left(1+\ln\left(\frac{\chi\, q^{3/4}}{4\eta^4\, q^{1/2}}\right)+\widetilde{O}(q^{-1/8})\right)\left(\left(4\eta^4-\frac{\chi^2}{4}\right)q^{1/2}+\widetilde{O}(q^{3/8})\right)\right]\\
&=16+\frac{\chi^2\ln q}{16\eta^4}-\frac{\chi^2\ln(4\eta^4)}{4\eta^4}+\frac{\chi^2}{8\eta^4}+\frac{\chi^2\ln\chi}{4\eta^4}\\
&+\left(1-\frac{\chi^2}{16\eta^4}\right)\left(4-4\ln(4\eta^4)+\ln q+4\ln\chi\right)+\widetilde{O}(q^{-1/8})\\
&=20-\frac{\chi^2}{8\eta^4}-4\ln(4\eta^4)+\ln q+4\ln\chi+\widetilde{O}(q^{-1/8})\,.
\end{aligned}$$

$\square$

The computational complexity of the relation search step is given by the number of factor base pairs, which is roughly $8\eta^8\, q$, multiplied by the computational cost of processing one such pair, as was discussed in the beginning of Section 3.3. The complexity of the linear algebra step depends quadratically on $\#\mathcal{F}_\chi$ and linearly on the average row weight given by Theorem 3.4.1. The total memory cost is given by the size of the graph, which is $\chi\, q^{3/4}$, multiplied by the memory cost of storing one vertex. For a given q the largest meaningful choice for χ is λ, given by Theorem 3.2.1, since at that point all of the required full relations have been found and there is nothing more to do.

Remark 3.4.2. In fact, it is easy to see that η, as a function of χ, has a global minimum at a point where $\chi = 4\eta^2$. Substituting this into (3.4.1) yields precisely the defining equation of λ.

On the other hand, the index calculus attack is no faster than Pollard rho unless $8\eta^8\, q \ll q^{3/2}$, which yields the upper bound $\eta \ll 2^{-3/8}\, q^{1/16}$. We denote the χ corresponding to $\eta = 2^{-3/8}\, q^{1/16}$ by $\chi_{\min}$. Hence we assume that the smallest meaningful size for the graph is $\chi_{\min}\, q^{3/4}$.

For any fixed q, the value of η increases as χ decreases from λ towards $\chi_{\min}$, but what is interesting is the rate of change of η. It turns out that η increases quite slowly until χ gets close to 1. In Figure 3.1 we show the ratios of both the relation search complexity and linear algebra complexity relative to those of Diem's algorithm, as functions of χ, for several field sizes.

Another point of view is to look at the complexities of relation search and linear algebra relative to the best possible ones, i.e. those obtained in the case of unrestricted memory ($\chi = 4\lambda^2$), as functions of memory cost relative to $N_{\max}$. These are presented in Figure 3.2, again for several field sizes. These graphs are instructive since we already have presented the values for the fastest case ($\chi = \lambda$) in Tables 3.1 and 3.3. For example, consider the field size $q = 2^{60}$ and $\chi/(4\lambda^2) = 1/5$. Then $(\eta/\lambda)^8 \approx 2.75$, so the running time is 2.75 times that of the running time in the unrestricted case, but the memory use has been cut down to a fifth of the original. We get more concrete numbers by scaling the results of Table 3.1. The memory use is 10098 TB and the computational complexity is approximately $q^{1.226} = 2^{73.589}$ field multiplications.

Yet a third point of view is to compare our algorithm to Pollard rho when memory use is even more restricted. Figure 3.3 shows the ratio of the complexities for several field sizes as a function of χ. The key point to observe is that one can do significantly better than Pollard rho even if χ is taken to be very small. So if one has massive amounts of computing power available the best approach might be to choose some suitable $\chi \ll 1$, although the memory cost is still going to be extremely large. For example, if we take $q = 2^{70}$ and $\chi = 2^{-6} \approx 0.016$, the graph will end up containing $\chi\, q^{3/4} = 2^{46.5} \approx 10^{14.0}$ vertices. Using the same estimate as in Section 3.3, this corresponds to 67300 TB of memory. The complexity of relation search will be $8\eta^8\, q \approx q^{1.386}$. As in Section 3.3, we can use the value $7\log_2 q + 13$ for the number of field multiplications needed to process one pair of factor base elements, so the total number of field multiplications can be estimated to be $(7\log_2 q + 13) \cdot 8\eta^8\, q \approx q^{1.514}$. Of course this is bigger than $q^{3/2}$, but the unit of time in Pollard rho and other generic algorithms involves operations in the Jacobian, and for non-hyperelliptic genus 3 curves each group operation costs around 130–185 field multiplications plus 2 field inversions [FOR].

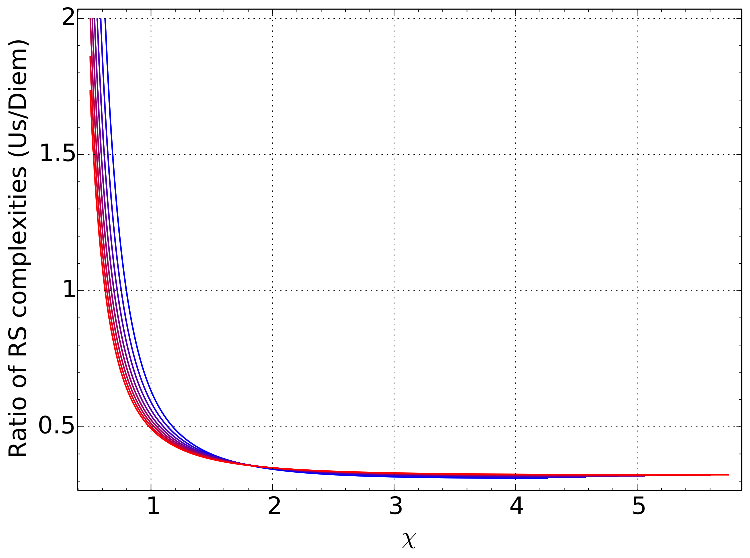
Recall that in Diem's algorithm (see [Die2]) the size of the factor base should be taken to be at least

$$\#\mathcal{F}_{\mathrm{Diem}} := \lceil ((3/2)\ln q + 4)^{1/2}\, q^{1/2} \rceil$$

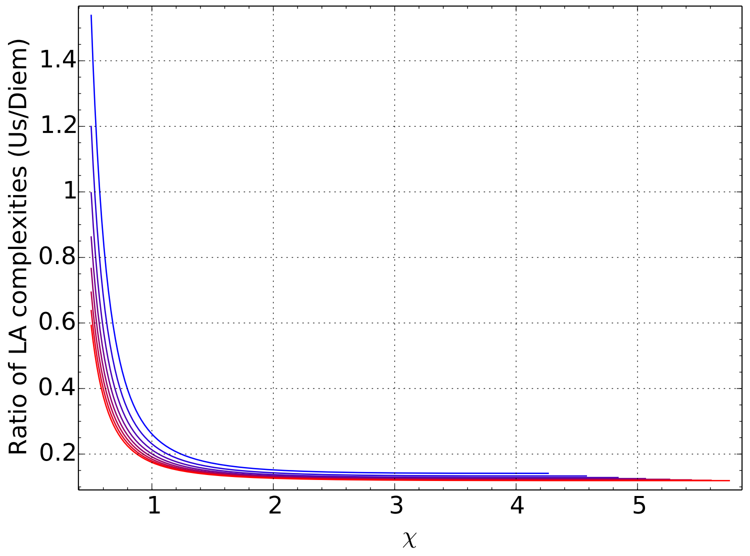
whereas the size of our factor base is $\#\mathcal{F}_\chi = 4\eta^4\, q^{1/2}$. Using the defining equation of η to

Figure 3.1: Complexities relative to those of Diem's algorithm, for field sizes $q = 2^{60}, 2^{80}, 2^{100}, 2^{120}, 2^{140}, 2^{160}, 2^{180}, 2^{200}$

(a) Relation search (RS)



(b) Linear algebra (LA)

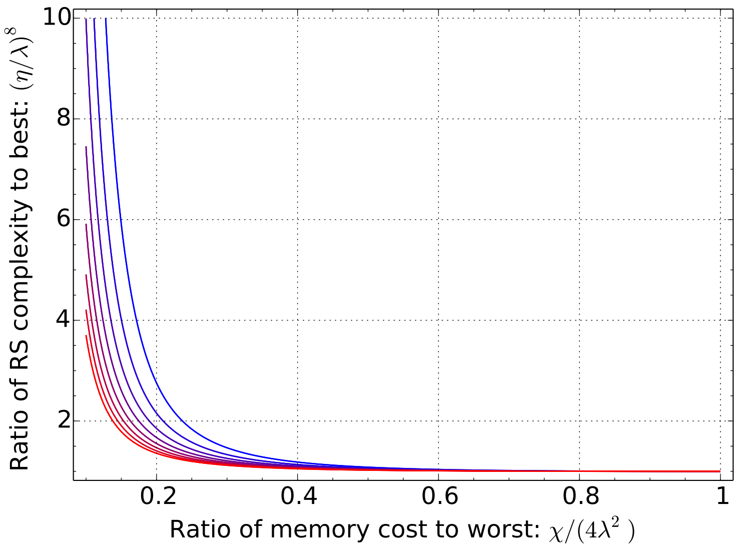


write

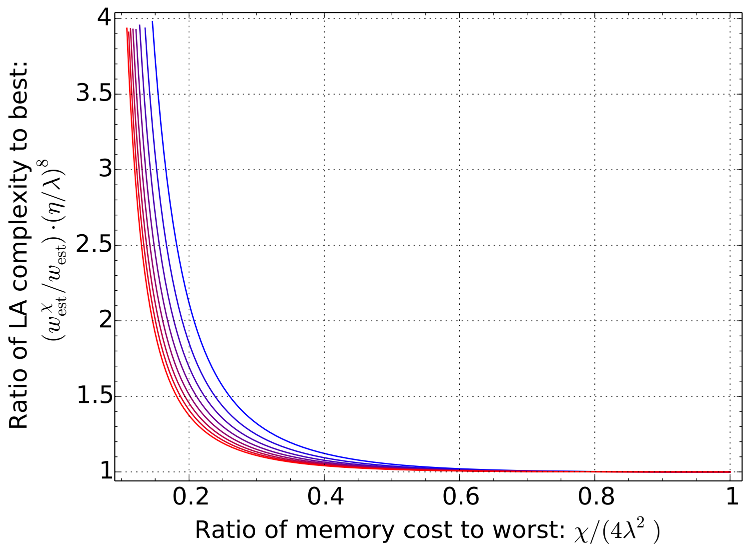
$$\ln q = 32\eta^8 - \frac{32\eta^4}{\chi^2} + 2 + 8\ln(2\eta^2) - 4\ln\chi$$

Figure 3.2: Complexities relative to the case of unrestricted memory, for field sizes $q = 2^{60}, 2^{80}, 2^{100}, 2^{120}, 2^{140}, 2^{160}, 2^{180}, 2^{200}$

(a) Relation search (RS)



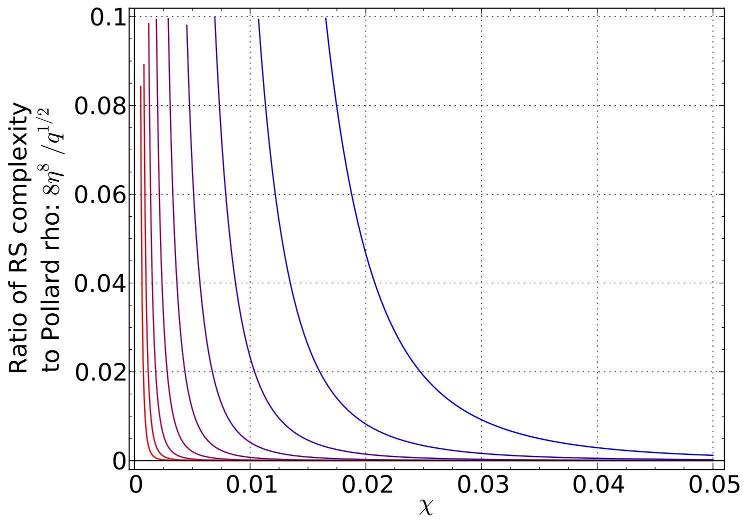
(b) Linear algebra (LA)



we find that

$$\rho_{\text{RS}}^{\chi}(q) := \left(\frac{\#\mathcal{F}_\chi}{\#\mathcal{F}_{\text{Diem}}}\right)^2 = \frac{1}{3}\left(1 - \frac{1}{\eta^4\chi^2} + \frac{1}{16\eta^8} + \frac{\ln(2\eta^2)}{4\eta^8} - \frac{\ln\chi}{8\eta^8} + \frac{1}{24\eta^8}\right)^{-1}$$

Figure 3.3: Relation search complexity relative to Pollard rho, for field sizes $q = 2^{60}, 2^{65}, 2^{70}, 2^{75}, 2^{80}, 2^{85}, 2^{90}, 2^{95}, 2^{100}$



which approaches 1/3 as $q \to \infty$ no matter what the fixed number χ is. This is the ratio plotted in Figure 3.1 on the left-hand side. Note that both in Diem's algorithm and in our variant the size of the factor base is chosen to be minimal so that all pairs of factor base elements are being used. So if we assume that in both algorithms the complexity of processing a pair of factor base elements is the same, the ratio $\rho_{\mathrm{RS}}^{\chi}$ measures the ratio of the complexities of the relation search steps.

Similarly, we define

$$\rho_{\mathrm{LA}}^{\chi}(q) := \frac{20 - \chi^2/(8\eta^4) - 4\ln(4\eta^4) + \ln q + 4\ln\chi}{6 + 3\ln q} \left(\frac{\#\mathcal{F}_\chi}{\#\mathcal{F}_{\mathrm{Diem}}}\right)^2 .$$

The ratio $\rho_{\mathrm{LA}}^{\chi}$ approaches 1/9 as $q \to \infty$ and it is the ratio plotted in Figure 3.1 on the right-hand side.

Lemma 3.4.3. *For any fixed χ, asymptotically as $q \to \infty$ our algorithm is 3 times faster than Diem's algorithm for relation search and 9 times faster for linear algebra.* □

Remark 3.4.4. As we have pointed out earlier, in our algorithm it is possible to take the factor base to be slightly smaller and re-run the graph building/relation searching step once it has failed for the first time. At that point the graph is huge and it should not take long to find the missing full relations. This is not possible in Diem's approach. We also save time because we only need to find roughly $q^{1/8}$ random points on the curve compared to over $q^{1/2}$ which have to be found in Diem's algorithm. Of course we then need to compute more intersection divisors to find the remaining factor base elements, but all of that work also builds the graph.

Remark 3.4.5. There is another variant of Diem's algorithm, namely the *full algorithm* approach of Diem and Thomé [DT] (see [GTTD] for the hyperelliptic case). In this algorithm a very large graph is constructed, of the size $q^{5/6}$. Consequently the factor base can be chosen to be smaller, of the size $2q^{1/2}$. The graph is disconnected and is searched for cycles which are then used to produce full relations. The complexity analysis of the *full algorithm* is difficult and the large graph size makes it impractical for all but the smallest examples so we will not discuss it further here.

3.5 Parallelization

In this section we study how the work of computing the DLP can be split between K computers. It turns out that with an increase in total computing time and total memory cost we can be split the computation between these K computers so that each needs to pay only a fraction of the original memory cost.

Suppose λ_K is a real number and we start by choosing a set $\mathcal{RP}^K$ of $4\lambda_K\, q^{1/8}$ random points on the curve and as usual use these to generate a factor base $\mathcal{F}^K$ of size $4\lambda_K^4\, q^{1/2}$. As usual at this point the graph will contain $4\lambda_K^4\, q^{1/2}$ triangles. Now distribute the entire factor base and the base vertices to each one of the K computers and split the triangles evenly between them so that each computer gets $4\lambda_K^4\, q^{1/2}/K$ triangles. Each computer starts building a graph and searching for full relations using their share of the triangles until they have each found approximately $4\lambda_K^4\, q^{1/2}$ full relations. Now all full relations are combined into a matrix and the linear algebra step is performed as usual. To simplify the notation, we denote λ_K simply by λ.

Theorem 3.5.1. *(Heuristic) If q is large enough and we take λ to be a root of*

$$\lambda \exp\left(4\lambda^8\right) = (K^2 q)^{1/8}$$

we can expect the algorithm to terminate successfully. Each of the K computers will end up with a graph of size

$$N_{max} = \frac{4\lambda^2\, q^{3/4}}{\sqrt{K}}\,.$$

The average row weight of the matrix will be approximately

$$18 - 8\ln\lambda + \ln(K^2 q)\,.$$

Proof. The proof is again similar to the proof of Theorem 3.2.1. We start with a set $\mathcal{RP}^K$ of $4\lambda\, q^{1/8} = \widetilde{O}(q^{1/8})$ random points and suppose that $N_{\max} = \widetilde{O}(q^{3/4})$. The size of the factor base and the number of vertices in the graph after the triangles have been constructed are

$$\#\mathcal{F}^K = 4\lambda^4\, q^{1/2} + \widetilde{O}(q^{3/8})\,, \qquad N_0 = 4\lambda^4\, q^{1/2} + \widetilde{O}(q^{3/8})\,.$$

Building the graphs from size N_0/K to $N_{\max}$ on each computer produces

$$\left(\frac{1}{2}+O(q^{-1/2})\right)\sum_{k=N_0/K}^{N_{\max}-1}\frac{k/q}{1-k/q}=\frac{N_{\max}^2}{4q}+\widetilde{O}(q^{1/4})$$

full relations. But each computer should produce $\#\mathcal{F}^K/K = 4\lambda^4\, q^{1/2}/K+\widetilde{O}(q^{3/8})$ full relations, so we get an equation and solve

$$N_{\max}=\frac{4\lambda^2\, q^{3/4}}{\sqrt{K}}\left(1+\widetilde{O}(q^{-1/8})\right)=\frac{4\lambda^2\, q^{3/4}}{\sqrt{K}}+\widetilde{O}(q^{5/8})\,.$$

The number of pairs of factor base elements that each computer has is $\binom{\mathcal{F}^K}{2}=8\lambda^8\, q+\widetilde{O}(q^{7/8})$. We want this number to equal the number of pairs needed to build the graphs as explained above. Hence we need $8\lambda^8\, q+\widetilde{O}(q^{7/8})$ to equal

$$\left(1+O(q^{-1/2})\right)\sum_{k=N_0/K}^{N_{\max}-1}\frac{1}{(k/q)(1-k/q)}=q\ln\left(\frac{K^{1/2}q^{1/4}}{\lambda^2}\right)+\widetilde{O}(q^{7/8})$$

which gives the equation

$$\lambda\,\exp\left(4\lambda^8\right)=(K^2q)^{1/8}+\widetilde{O}(1)\,.$$

Finally, the average row weight is approximated from above by

$$16+\frac{2^2}{\#\mathcal{F}^K/K}\sum_{k=N_0/K}^{N_{\max}-1}\left(1+\ln\left(\frac{kK}{4\lambda^4\, q^{1/2}}\right)+\widetilde{O}(q^{-1/8})\right)\cdot\left(\frac{1}{2}+O(q^{-1/2})\right)\frac{k/q}{1-k/q}$$

$$=18-8\ln\lambda+\ln(K^2q)+\widetilde{O}(q^{-1/8})\,.$$

□

More generally, we can consider what happens if each of the K computers stops building their graphs when they reach size $\chi\, q^{3/4}/\sqrt{K}$, where χ is a fixed positive real number, and only proceed with relation search. Suppose $\eta_{K,\chi}$ is a real number and we start by choosing a set $\mathcal{RP}_\chi^K$ of $4\eta_{K,\chi}\, q^{1/8}$ random points on the curve. Then we get a result similar to Theorem 3.4.1. To simplify the notation, we denote $\eta_{K,\chi}$ by η.

Theorem 3.5.2. *(Heuristic) If q is large enough and we take η to be a root of*

$$2\eta^2\,\exp\left(4\eta^8-4\eta^4/\chi^2+1/4\right)=\chi^{1/2}\,(K^2q)^{1/8}$$

we can expect the algorithm to terminate successfully. The average row weight of the matrix will be approximately

$$20-\frac{\chi^2}{8\eta^4}-4\ln(4\eta^4)+\ln(K^2q)+4\ln\chi\,.$$

Proof. Similar to the proofs of Theorem 3.4.1 and Theorem 3.5.1. □

Above we assumed that each one of the computers computes the same $8\eta^8 q$ intersection divisors but of course this is completely unnecessary if there is a very efficient way for the computers to communicate with each other. In this case each computer computes $\left(1/2 + O(q^{-1/2})\right) 8\eta^8 q/K$ intersection divisors and shares its results with the other $K-1$ computers. Unfortunately this is a massive amount of data to share. In fact, merely storing all these intersection divisors costs potentially much more memory than storing the graph since

$$\frac{4\eta^8 q}{K} \gg \frac{\chi q^{3/4}}{\sqrt{K}}$$

for practical field sizes unless K is impossibly large. Hence instead of storing the intersection divisors they should be streamed to all other computers as soon as they are generated and then deleted immediately afterwards. This speeds up the algorithm only if the connections between the computers are so fast that distributing the data over the network is faster than for each computer to generate it separately. We assume this is the case.

Let $M_{\text{Total}}^{K,\chi}$ be the total number of field multiplications needed per computer. We have

$$M_{\text{Total}}^{K,\chi} = M_{\text{pair}} \cdot \frac{1}{K} \binom{\#\mathcal{F}_\chi^K}{2} \approx (7\log_2 q + 13) \cdot \frac{8\eta_{K,\chi}^8 q}{K} .$$

Note that this is not exactly the same as $1/K$-th of the complexity in the unparallelized case because the coefficient $\eta_{K,\chi}$ is bigger, namely $\eta_{K,\chi}(q) = \eta_\chi(K^2 q)$.

The complexity of linear algebra is given by

$$\left(20 - \frac{\chi^2}{8\eta_{K,\chi}^4} - 4\ln(4\eta_{K,\chi}^4) + \ln(K^2 q) + 4\ln\chi\right) \cdot 16\eta_{K,\chi}^8 q ,$$

which is worse than the complexity in the unparallelized case, again due to $\eta_{K,\chi}(q) = \eta_\chi(K^2 q)$.

We demonstrate these results in the case of unrestricted memory, i.e. when $\chi = 4\lambda_K^2$. The ratio of complexities of the linear algebra steps in the parallelized and unparallelized cases is

$$\rho_{\text{LA}}^K := \frac{18 - 8\ln\lambda_K + \ln(K^2 q)}{18 - 8\ln\lambda + \ln q} \left(\frac{\lambda_K}{\lambda}\right)^8 .$$

We denote

$$\text{MPC}_K \text{ (Memory cost Per Computer)} := \frac{4\lambda_K^2 q^{3/4}}{\sqrt{K}} \cdot (\lceil(\log_2 q)/64\rceil \cdot 370 \text{ bytes})$$

and

$$\text{IDPC}_K \text{ (Intersection Divisors Per Computer with } \mathbb{F}_q\text{-rational points)} := \frac{4\lambda_K^8 q}{K} ,$$

the latter measuring the amount of data that needs to be shared.

These numbers for a few practical field sizes are shown in Tables 3.4, 3.5, 3.6. It is also easy to see how to scale the values in Tables 3.4, 3.5, 3.6 to obtain corresponding numbers in the case $\chi < 4\lambda_K^2$. For example, to find $M_{\text{Total}}^{K,\chi}$, simply scale the value of M_{Total}^{K} by the coefficient given by Figure 3.2 for a field of size K^2q. To find $\text{MPC}_{K,\chi}$, scale the value of MPC_K by $(\eta_{K,\chi}/\lambda_K)^2$. Concretely, consider e.g. $q = 2^{60}$, $K = 100$ and $\chi/(4\lambda_K^2) = 1/50$. Then

$$\log_2 M_{\text{Total}}^{K,\chi} = \log_2 \left[M_{\text{Total}}^{K} \cdot \left(\frac{\eta_{K,\chi}}{\lambda_K} \right)^8 \right] = 79.03607$$

and $\text{MPC}_{K,\chi} = 106$ TB.

In the above our main concern was with reducing the per computer memory cost. One should keep in mind that parallelization makes the complexity of the linear algebra part slightly worse and for large K the difference between the relation search step and the linear algebra step becomes very large, unless χ is taken to be smaller. To make linear algebra faster, each computer could perform some preprocessing of the full relations before they are combined into the full linear algebra problem. There are also several ways of parallelizing relation search within the K memory units to further reduce the complexity $M_{\text{Total}}^{K,\chi}$. For instance, each processor connected to a memory unit can compute its own share of intersection divisors which are then collected together and used to build the local graph.

Table 3.4: Parallelization: $K = 4$

Field size q	$\log_2 M_{\text{Total}}$	ρ_{LA}	MPC	$\log_2 \text{IDPC}$
2^{60}	70.21903	1.11420	25648 TB	60.46080
2^{70}	80.64203	1.09991	$9.62 \cdot 10^{6}$ TB	70.66761
2^{80}	91.01129	1.08883	$1.80 \cdot 10^{9}$ TB	80.84890
2^{90}	101.3389	1.07996	$3.35 \cdot 10^{11}$ TB	91.01023
2^{100}	111.63331	1.07272	$6.21 \cdot 10^{13}$ TB	101.15555
2^{110}	121.90060	1.06667	$1.15 \cdot 10^{16}$ TB	111.28773
2^{120}	132.14535	1.06156	$2.13 \cdot 10^{18}$ TB	121.40894
2^{140}	152.58044	1.05338	$1.09 \cdot 10^{23}$ TB	141.62479
2^{160}	172.95868	1.04712	$3.67 \cdot 10^{27}$ TB	161.81275
2^{180}	193.29321	1.04217	$1.24 \cdot 10^{32}$ TB	181.97919
2^{200}	213.59307	1.03816	$5.56 \cdot 10^{36}$ TB	202.12853

Table 3.5: Parallelization: $K = 100$

Field size q	$\log_2 M_{\text{Total}}$	ρ_{LA}	MPC	$\log_2 \text{IDPC}$
2^{60}	65.76817	1.40309	5304 TB	56.00995
2^{70}	76.16726	1.35025	$1.98 \cdot 10^{6}$ TB	66.19284
2^{80}	86.51785	1.30967	$3.69 \cdot 10^{8}$ TB	76.35545
2^{90}	96.83049	1.27752	$6.85 \cdot 10^{10}$ TB	86.50181
2^{100}	107.11261	1.25142	$1.27 \cdot 10^{13}$ TB	96.63485
2^{110}	117.36965	1.22982	$2.35 \cdot 10^{15}$ TB	106.75678
2^{120}	127.60571	1.21162	$4.33 \cdot 10^{17}$ TB	116.86931
2^{140}	148.02689	1.18270	$2.20 \cdot 10^{22}$ TB	137.07124
2^{160}	168.39449	1.16072	$7.45 \cdot 10^{26}$ TB	157.24856
2^{180}	188.72061	1.14346	$2.51 \cdot 10^{31}$ TB	177.40660
2^{200}	209.01367	1.12954	$1.12 \cdot 10^{36}$ TB	197.54912

Table 3.6: Parallelization: $K = 1000$

Field size q	$\log_2 M_{\text{Total}}$	ρ_{LA}	MPC	$\log_2 \text{IDPC}$
2^{60}	62.57026	1.63014	1714 TB	52.81204
2^{70}	72.95535	1.54507	$6.39 \cdot 10^{5}$ TB	62.98094
2^{80}	83.29477	1.48017	$1.19 \cdot 10^{8}$ TB	73.13238
2^{90}	93.59828	1.42905	$2.20 \cdot 10^{10}$ TB	83.26961
2^{100}	103.87281	1.38774	$4.07 \cdot 10^{12}$ TB	93.39505
2^{110}	114.12344	1.35366	$7.52 \cdot 10^{14}$ TB	103.51057
2^{120}	124.35401	1.32508	$1.39 \cdot 10^{17}$ TB	113.61761
2^{140}	144.76630	1.27981	$7.05 \cdot 10^{21}$ TB	133.81065
2^{160}	165.12699	1.24559	$2.38 \cdot 10^{26}$ TB	153.98106
2^{180}	185.44760	1.21881	$8.00 \cdot 10^{30}$ TB	174.13358
2^{200}	205.73615	1.19728	$3.58 \cdot 10^{35}$ TB	194.27161

Chapter 4

Avoiding Isogeny Attacks in Genus 3

4.1 Overview

Isogeny Attacks

Galbraith [Gal] showed that it was possible to attack elliptic curve cryptosystems by finding a weak curve in the isogeny class of the original curve. By *Tate's isogeny theorem* [Sil1] two elliptic curves E_1, E_2 over $\mathbb{F}_q$ are isogenous if $E_1(\mathbb{F}_p)$ and $E_2(\mathbb{F}_p)$ have the same number of points. Since point counting can be done in logarithmic time using the *Schoof-Elkies-Atkin algorithm*, it is easy to check whether two elliptic curves are isogenous. Next Galbraith used the volcano structure of ℓ-isogeny graphs found by Kohel [Koh] and the CM theory of elliptic curves [Sil2] to construct an explicit isogeny between the two curves. The discrete logarithm can now be mapped from the original curve to the weak curve, where it can possibly be solved. A similar attack against hyperelliptic curves of genus 2 is possible using recently developed algorithms for computing explicit isogenies [Wes, Dud] but is less straightforward due to the significantly more complicated structure of the isogeny graphs.

In genus 3 things are radically different due to most genus 3 curves being non-hyperelliptic and thus vulnerable to the index calculus attack of Chapter 3, which means that finding a weak Jacobian in the isogeny class should be very easy. Indeed, we briefly explained already in Section 2.6 how it is possible to attack the DLP on a hyperelliptic Jacobian by constructing an arbitrary explicit isogeny to an isogenous principally polarized 3-dimensional abelian variety, which according to Theorems 2.6.6, 2.6.7, 2.6.1 will almost certainly be isomorphic to the Jacobian of a non-hyperelliptic genus 3 curve, although possibly only over a quadratic extension field [ZLR]. Once the DLP is mapped to this non-hyperelliptic Jacobian, all that remains to be done is use the $\widetilde{O}(p)$ non-hyperelliptic index calculus of Chapter 3.

Remark 4.1.1. As was mentioned above, the target abelian variety might be isomorphic to a Jacobian only over a quadratic extension field. It is clear from Chapter 3 that the attack becomes impossibly costly if this is the case (see e.g. Theorem 3.2.1 and Table 3.1). Instead, to succeed the attacker must be able to recover the Jacobian structure over the original field, which happens with probability approximately 1/2 (see [ZLR]).

Definition 4.1.2. Let $\mathcal{A}$ be a g-dimensional abelian variety. A subgroup $\mathcal{K} \subseteq \mathcal{A}[\ell]$ is said to be *isotropic* if the ℓ-Weil pairing becomes trivial when restricted to $\mathcal{K}$. The isotropic subgroup $\mathcal{K}$ is *maximal isotropic* if it is not a subgroup of any strictly larger isotropic subgroup. When ℓ is prime, every maximal isotropic subgroup is isomorphic to $(\mathbb{Z}/\ell\mathbb{Z})^g$.

Definition 4.1.3. Let $\mathcal{A}$ be 3-dimensional. An isogeny with maximal isotropic kernel $\mathcal{K} \subset \mathcal{A}[\ell]$ isomorphic to $(\mathbb{Z}/\ell\mathbb{Z})^3$ is called an *(ℓ, ℓ, ℓ)-isogeny.*

Fortunately, computing isogenies between 3-dimensional Jacobians seems to be very difficult in general. From the point of view of the attacker, the best types of isogenies for this purpose seem to be (ℓ, ℓ, ℓ)-isogenies and *cyclic isogenies*, i.e. isogenies with a cyclic kernel. Of these two (ℓ, ℓ, ℓ)-isogenies are significantly easier to compute, essentially because they induce a natural principal polarization on the target abelian variety.

The first demonstration of an isogeny attack in genus 3 was given by Smith in [Smi]. His algorithm gives a very efficient way of computing explicitly certain $(2, 2, 2)$-isogenies. Although the algorithm is very fast, it only works for approximately 18.57% of all hyperelliptic curves over a given finite field. Thus one can easily avoid this attack by using curves such that the conditions necessary for the algorithm to work are not met, e.g. a curve whose hyperelliptic polynomial factors over $\mathbb{F}_p$ in a particular way (see [Smi]).

A more general but also significantly more complicated method for computing (ℓ, ℓ, ℓ)-isogenies for any ℓ is developed and discussed in references [LR, CR, Rob]. This method has so far only been implemented for Jacobians of genus 2 curves, but there does not seem to be any reason why it would not work also in genus 3. Consequently one should assume (ℓ, ℓ, ℓ)-isogenies to be possible to compute explicitly for any ℓ not too large.

Computing cyclic isogenies is significantly more difficult because they do not respect principal polarizations. This means that one needs to somehow recover an appropriate principal polarization on the target abelian variety. It is not clear how this could be done, although one possible approach is outlined in [Rob2] and further developed in the case of genus 2 by Dudeanu in [Dud]. The method of Dudeanu places extremely strong restrictions on the rings of real multiplication of the source and the target Jacobians, and has no obvious generalization to the case of genus 3. Nevertheless, it is not unreasonable to assume that in the near future also cyclic isogenies between Jacobians of genus 3 curves can be computed. There are currently no methods known for computing any other types of isogenies.

Constructing Genus 3 Curves

We already explained in Section 2.6 how the genus 3 CM method of Weng [Wen] can be used to construct hyperelliptic genus 3 curves with a predetermined number of points. This involved choosing a sextic CM field $K = K_0(i)$, where K_0 is a totally real cubic number field. One of the main difficulties in the genus 3 case is that for a randomly chosen CM field the resulting curve will be non-hyperelliptic with overwhelming probability due to the hyperelliptic locus having codimension 1 in the moduli space of genus 3 curves (Theorem 2.6.6). Weng solved this problem by taking the CM to field to be such that $i \in \mathcal{O}_K$, which implies by an extension of Torelli's theorem that the curve must be hyperelliptic (see [Wen] and Theorem 4.4.2).

Remark 4.1.4. The genus 3 CM method of Weng produces very special hyperelliptic curves with an automorphism of order 4. No analogous algorithm for constructing more general hyperelliptic genus 3 curves is known.

For cryptographic purposes we need to choose K/K_0 and a p-Weil number $\pi \in \mathcal{O}_K$ so that one of $\mathrm{Nm}(i\pi \mp 1)$ is almost prime. Recall from Section 2.6 that these norms compute the orders of the groups $\{\mathrm{Jac}_{C_i}(\mathbb{F}_p)\}_i$ for the four quartic twists $\{C_i\}$ of a hyperelliptic curve over $\mathbb{F}_p$. In this Chapter we discuss a way of doing this which allows us to both compute and to control the index $[\mathcal{O}_K : \mathbb{Z}[\pi, \overline{\pi}]]$. This is desirable because if $\mathcal{A} \to \mathcal{B}$ is an isogeny with kernel $\mathcal{K} \subseteq \mathcal{A}[\ell^n]$ for some prime ℓ, and if

$$\mathrm{End}\,\mathcal{A} = \mathcal{O}_K\,, \qquad \mathrm{End}\,\mathcal{B} = \mathcal{O}\,,$$

then $[\mathcal{O}_K : \mathcal{O}]$ must divide ℓ^{5n}. On the other hand $[\mathcal{O}_K : \mathcal{O}]$ divides $[\mathcal{O}_K : \mathbb{Z}[\pi, \overline{\pi}]]$ so ℓ must be a divisor of $[\mathcal{O}_K : \mathbb{Z}[\pi, \overline{\pi}]]$. We will force $[\mathcal{O}_K : \mathbb{Z}[\pi, \overline{\pi}]]$ to be divisible only by 2 and large primes[1]. It is not feasible to compute isogenies whose kernels are contained in large torsion subgroups, so it suffices to worry about isogenies with kernel contained in power-of-2 torsion subgroups of $\mathcal{A}$. In our construction of K/K_0 and π the attacker will only be able to reach the Jacobian of a non-hyperelliptic curve if the kernel of the isogeny contains points of high power-of-2 order. The connection to endomorphism rings is given by Theorem 4.4.2, namely the endomorphism ring of the target must be an order $\mathcal{O} \supset \mathbb{Z}[\pi, \overline{\pi}]$ such that $i \notin \mathcal{O}$. Computing such isogenies in one step is impossible since the coordinates of the points in the kernel would have to be represented using huge extension fields. In practice the attacker would have to compute a long enough chain of isogenies with smaller $\mathbb{F}_p$-rational kernels[2], e.g. $\mathbb{F}_p$-rational $(2,2,2)$-isogenies. We will also give a heuristic reason for why the curves we construct should be secure against isogeny attacks by cyclic isogenies unless remarkable progress in the theory of explicit isogeny computations is made. If the curve is now carefully selected so that its Jacobian admits no chain of isogenies with $\mathbb{F}_p$-rational maximal isotropic kernels contained in power-of-2 torsion leading to a Jacobian of a non-hyperelliptic curve[3], one can assume the curve to be relatively safe against isogeny attacks.

[1]This is not quite true. The index can also be divisible by certain small primes but they end up being less relevant for the attacker.

[2]It is necessary that the kernels are defined over $\mathbb{F}_p$. Otherwise the attacker will end up at an abelian variety defined over an extension field and the index calculus algorithm of Chapter 3 becomes unwieldy. One could imagine combining such an isogeny with another isogeny so that the target eventually has a model over $\mathbb{F}_p$, but it is not clear how this could be done.

[3]In particular, it should not admit any such chains of $(2,2,2)$-isogenies, which are the easiest to compute.

4.2 A Family of Sextic CM Fields

Requirements

Let $K_0 = \mathbb{Q}(w)$ be a totally real cubic number field and $K = K_0(i)$ a sextic CM field. We want to find rational primes p such that $p = \pi\overline{\pi}$, where $\pi \in \mathcal{O}_K$ is a p-Weil number and $K = \mathbb{Q}(\pi)$. We then construct a hyperelliptic genus 3 curve C using the genus 3 CM method of Weng [Wen] (recall Section 2.6) such that the Jacobian Jac_C is a simple ordinary 3-dimensional abelian variety and $\mathrm{Jac}_C(\mathbb{F}_p)$ has order $\mathrm{Nm}(\pi - 1)$, which we want to be almost prime (divisible by a very large prime and possibly some powers of small primes).

Suppose the minimal polynomial of the totally real root w is $f(x) = x^3 + \alpha x^2 + \beta x + \gamma$ and that

$$\pi = a_0 + a_1 w + a_2 w^2 + i(b_0 + b_1 w + b_2 w^2)\,.$$

We then require that

$$\begin{aligned}\pi\overline{\pi} = &\left[a_0^2 + b_0^2 + \alpha\gamma(a_2^2 + b_2^2) - 2(a_1a_2 + b_1b_2)\gamma\right] \\ &+ \left[(\alpha\beta - \gamma)(a_2^2 + b_2^2) + 2(a_0a_1 + b_0b_1) - 2(a_1a_2 + b_1b_2)\beta\right] w \\ &+ \left[a_1^2 + b_1^2 + (a_2^2 + b_2^2)(\alpha^2 - \beta) + 2(a_0a_2 + b_0b_2) - 2(a_1a_2 + b_1b_2)\alpha\right] w^2\end{aligned}$$

equals a prime p, which results in three conditions:

$$a_0^2 + b_0^2 + \alpha\gamma(a_2^2 + b_2^2) - 2(a_1a_2 + b_1b_2)\gamma = p\,,$$

$$(\alpha\beta - \gamma)(a_2^2 + b_2^2) + 2(a_0a_1 + b_0b_1) - 2(a_1a_2 + b_1b_2)\beta = 0\,,$$

$$a_1^2 + b_1^2 + (a_2^2 + b_2^2)(\alpha^2 - \beta) + 2(a_0a_2 + b_0b_2) - 2(a_1a_2 + b_1b_2)\alpha = 0\,.$$

In addition, the following conditions must be met:

(1) The polynomial $f(x) = x^3 + \alpha x^2 + \beta x + \gamma$ must be irreducible.

(2) π generates the sextic CM field $K/\mathbb{Q}$.

(3) The number p must be prime.

(4) Some of the parameters must be large enough so that p can be large.

(5) All complex embeddings of π must have absolute value $p^{1/2}$.

(6) $\pi + \overline{\pi}$ must be relative prime to p, i.e. π must be an ordinary Weil number [MW].

(7) K_0 must have class number 1 for the CM method to work.

(8) K must have small class number for the CM method to work.

(9) The index $[\mathcal{O}_{K_0} : \mathbb{Z}[w]]$ should be small and preferably 1, since this simplifies the CM method.

(10) The index $[\mathcal{O}_K : \mathbb{Z}[\pi, \overline{\pi}]]$ must be computable and controllable.

(11) The number $\mathrm{Nm}(\pi - 1)$ must be almost prime.

(12) It must be easy to generate several such sets of parameters.

Remark 4.2.1. Due to Cohen-Martinet type heuristics (see [CM]) there is a good chance that the totally real field K_0 has class number 1, but in general the field K will have huge class number unless the discriminant of $f(x)$ is small, which will be likely to happen only when α, β, γ are small.

A Family of Sextic CM Fields

We now show one way of choosing sets of parameters satisfying the twelve conditions listed above. Algorithm 4.2.1 attempts to construct a sextic CM field $K/\mathbb{Q}$ with totally real subfield K_0, together with an ordinary p-Weil number $\pi \in \mathcal{O}_K$ such that $\pi\overline{\pi} = p$ is a large prime and that a 3-dimensional principally polarized abelian variety $\mathcal{A}/\mathbb{F}_p$ with CM by $\mathcal{O}_K$ and π acting as the Frobenius endomorphism has $\#\mathcal{A}(\mathbb{F}_p)$ almost prime.

If Algorithm 4.2.1 returns FAIL, simply restart it with new input parameters. In Section 4.3 we explain how the input parameters should be chosen so that a successful output can be expected with reasonable probability, and give a revised version of Algorithm 4.2.1 implementing these strategies (see Algorithms 4.3.1,4.3.2). For now, we assume that Algorithm 4.2.1 successfully outputs a tuple $(\alpha, \beta, \gamma, A, B, k, a_0, p, Q, K_0, K, P, C, \pi, \# \mathrm{Cl}(K))$ and prove some facts about it.

Lemma 4.2.2. *The index* $[\mathcal{O}_K : \mathbb{Z}[i, w]]$ *divides* $2^5[\mathcal{O}_{K_0} : \mathbb{Z}[w]]^2$.

Proof. It is a standard result in algebraic number theory that if $K/\mathbb{Q}$ and $L/\mathbb{Q}$ are number fields such that $[K \cdot L : \mathbb{Q}] = [K : \mathbb{Q}][L : \mathbb{Q}]$ and $d = \gcd(\Delta(\mathcal{O}_K/\mathbb{Z}), \Delta(\mathcal{O}_L/\mathbb{Z}))$, then

$$\mathcal{O}_{K\cdot L} \subseteq \frac{1}{d}\mathcal{O}_K \cdot \mathcal{O}_L \,.$$

Compute

$$d = \gcd\left(\Delta(\mathcal{O}_{K_0}/\mathbb{Z}), \Delta(\mathcal{O}_{\mathbb{Q}(i)}/\mathbb{Z})\right) \,.$$

This is either 1, 2 or 4 since $\Delta(\mathcal{O}_{\mathbb{Q}(i)}/\mathbb{Z}) = 4$. In any case,

$$\mathcal{O}_K \subseteq \mathbb{Z} + \frac{1}{4}\left(\mathbb{Z}\eta_1 + \mathbb{Z}\eta_2 + \mathbb{Z}i + \mathbb{Z}i\eta_1 + \mathbb{Z}i\eta_2\right) \,,$$

where $\mathcal{O}_{K_0} = \mathbb{Z} + \mathbb{Z}\eta_1 + \mathbb{Z}\eta_2$. Take any element $\eta \in \mathcal{O}_K$ and write it as

$$\eta = a + \frac{b\eta_1 + c\eta_2 + di + ei\eta_1 + fi\eta_2}{4} \,.$$

Algorithm 4.2.1 Constructing a sextic CM field

```
1: **Input:** $\alpha, \beta$ integers, $k, B$ positive integers and $A$ a power of 2 such that $A^2 \mid 2B$.
2: Let $\gamma := \alpha\beta - 2B/A^2$.
3: Let $a_0 := 2^{3k-2}(2B^2/A) + 2^{k-1}A(\alpha^2 - \beta)$
4: Let $p := a_0^2 + 2^{2k}A^2\alpha\gamma + 1$.
5: **if** $p$ is not prime **then**
6:     **Output:** FAIL
7: **end if**
8: Let $Q := 1 + 2^{2k}B\alpha + 2^{4k}B^2\beta + 2^{6k}B^3\gamma$.
9: **if** $Q$ has small prime divisors **then**
10:     **Output:** FAIL
11: **end if**
12: Let $f(x) := x^3 + \alpha x^2 + \beta x + \gamma$.
13: **if** $f$ is not irreducible over $\mathbb{Q}$ **then**
14:     **Output:** FAIL
15: **end if**
16: Let $\Delta := \alpha^2\beta^2 - 4\beta^3 - 4\alpha^3\gamma - 27\gamma^2 + 18\alpha\beta\gamma$.
17: **if** $\Delta \leq 0$ **then**
18:     **Output:** FAIL
19: **end if**
20: Let $K_0$ be the field generated by a root $w$ of $f(x)$.
21: **if** $[\mathcal{O}_{K_0} : \mathbb{Z}[w]]$ is not small **then**
22:     **Output:** FAIL
23: **end if**
24: Let $K := K_0(i)$.
25: **if** $p$ is ramified in $K$ **then**
26:     **Output:** FAIL
27: **end if**
28: Let $\Pi := a_0 - 2^k A w^2 + i(1 - 2^{2k}Bw)$.
29: Compute $N_\pm := \mathrm{Nm}(\pm\Pi - 1)$.
30: **if** $N_s$ equals a large prime times a small cofactor $C_s$ for a sign $s$ **then**
31:     Let $P := N_s/C_s$, $C := C_s$ and $\pi := s\Pi$.
32: **else**
33:     **Output:** FAIL
34: **end if**
35: **if** $|\varphi(\pi + \overline{\pi})| > 2p^{1/2}$ for some embedding $\varphi : \mathcal{O}_{K_0} \hookrightarrow \mathbb{R}$ **then**
36:     **Output:** FAIL
37: **end if**
38: **if** the class number of $K_0$ is not 1 **then**
39:     **Output:** FAIL
40: **end if**
41: Compute the class number $\#\,\mathrm{Cl}(K)$ of $K$.
42: **Output:** $(\alpha, \beta, \gamma, A, B, k, a_0, p, Q, K_0, K, P, C, \pi, \#\,\mathrm{Cl}(K))$
```

The relative trace of η is

$$\operatorname{Tr}_{K/K_0}\eta = 2a + \frac{b\eta_1 + c\eta_2}{2} \in \mathcal{O}_{K_0}\,,$$

so b and c must be even. Similarly computing $\operatorname{Tr}_{K/K_0} i\eta \in \mathcal{O}_{K_0}$ shows that d, e, f are even. Hence

$$\mathcal{O}_K \subseteq \mathbb{Z} + \frac{1}{2}\left(\mathbb{Z}\eta_1 + \mathbb{Z}\eta_2 + \mathbb{Z}i\eta_1 + \mathbb{Z}i\eta_2\right)\,. \tag{4.2.1}$$

Let M be the $\mathbb{Z}$-module $\mathbb{Z}\eta_1 + \mathbb{Z}\eta_2 + \mathbb{Z}i + \mathbb{Z}i\eta_1 + \mathbb{Z}i\eta_2$. From (4.2.1) we see that

$$[\mathbb{Z} + (1/2)M : \mathcal{O}_K][\mathcal{O}_K : \mathbb{Z} + M] = [\mathbb{Z} + (1/2)M : \mathbb{Z} + M] = 2^5\,,$$

so $[\mathcal{O}_K : \mathbb{Z} + M] \mid 2^5$. Next observe that

$$[\mathcal{O}_K : \mathbb{Z} + M][\mathbb{Z} + M : \mathbb{Z}[i, w]] = [\mathcal{O}_K : \mathbb{Z}[i, w]]\,,$$

but $[\mathbb{Z} + M : \mathbb{Z}[i, w]] = [\mathcal{O}_{K_0} : \mathbb{Z}[w]]^2$ so $[\mathcal{O}_K : \mathbb{Z}[i, w]] \mid 2^5[\mathcal{O}_{K_0} : \mathbb{Z}[w]]^2$. □

Theorem 4.2.3. *If Algorithm 4.2.1 returns successfully, it produces a sextic CM field K with real subfield K_0, a prime p and an ordinary p-Weil number π such that*

$$[\mathcal{O}_K : \mathbb{Z}[\pi, \overline{\pi}]] \mid 2^{6k+25}A^5B^2Q^3[\mathcal{O}_{K_0} : \mathbb{Z}[w]]^2\,.$$

More precisely, we have

$$2^{2k+3}Bw \in \mathbb{Z}[\pi, \overline{\pi}]\,, \quad 2^{k+1}Aw^2 \in \mathbb{Z}[\pi, \overline{\pi}]\,, \quad 2^4AQi \in \mathbb{Z}[\pi, \overline{\pi}]\,,$$

$$2^{2k+7}ABQiw \in \mathbb{Z}[\pi, \overline{\pi}]\,, \quad 2^{k+5}A^2Qiw^2 \in \mathbb{Z}[\pi, \overline{\pi}]\,, \quad [\mathcal{O}_K : \mathbb{Z}[i, w]] \mid 2^5[\mathcal{O}_{K_0} : \mathbb{Z}[w]]^2\,.$$

Proof. Showing that $\pi\overline{\pi} = p$ is a simple computation using

$$w^3 = -\alpha w^2 - \beta w - \gamma \quad \text{and} \quad w^4 = (\alpha^2 - \beta)w^2 + (\alpha\beta - \gamma)w + \alpha\gamma\,.$$

Let π' be any of the conjugates of π. Then π' satisfies the equation

$$x^2 - \kappa x + p = 0$$

for some (real) embedding κ of $\pi + \overline{\pi}$. The roots of this equation are $(\kappa \pm i\sqrt{4p - \kappa^2})/2$, where the square root is real by construction ($|\kappa| \leq 2p^{1/2}$). But both roots have absolute value $p^{1/2}$. This proves that π is a p-Weil number.

According to [MW] the p-Weil number π is ordinary if and only if $\pi + \overline{\pi}$ is relatively prime to p. If this was not the case, then some prime $\mathfrak{r} \subset \mathcal{O}_K$ divides both $(\pi)(\overline{\pi})$ and $(\pi + \overline{\pi})$, which implies that $\mathfrak{r}$ divides both $(\pi)^2$ and $(\overline{\pi})^2$, so it divides both (π) and $(\overline{\pi})$. But then $\mathfrak{r}^2$ divides (p), which is not the case by construction.

For the index, write

$$-(\Pi+\overline{\Pi})+2a_0 = 2^{k+1}Aw^2 \in \mathbb{Z}[\Pi,\overline{\Pi}]\,, \tag{4.2.2}$$

$$\Pi-\overline{\Pi} = 2i-2^{2k+1}Biw \in \mathbb{Z}[\Pi,\overline{\Pi}]\,. \tag{4.2.3}$$

Squaring (4.2.2) yields

$$2^{2k+2}A^2w^4 = 2^{2k+2}A^2(\alpha^2-\beta)w^2+2^{2k+2}A^2(\alpha\beta-\gamma)w+2^{2k+2}A^2\alpha\gamma \in \mathbb{Z}[\Pi,\overline{\Pi}]\,.$$

Using (4.2.2) and $(\alpha\beta-\gamma)A^2=2B$ this turns into

$$2^{2k+3}Bw \in \mathbb{Z}[\Pi,\overline{\Pi}]\,. \tag{4.2.4}$$

Next we need

$$2^{2k+3}Bw(2i-2^{2k+1}Biw) = 2^{2k+4}Biw-2^{4k+4}B^2iw^2 \in \mathbb{Z}[\Pi,\overline{\Pi}]\,. \tag{4.2.5}$$

Then compute

$$2^{4k+3}AB^2w^2(2i-2^{2k+1}Biw) = 2^{4k+4}AB^2(1+2^{2k}\alpha)iw^2$$
$$+2^{6k+4}AB^2\beta iw+2^{6k+4}AB^2\gamma i \in \mathbb{Z}[\Pi,\overline{\Pi}]\,.$$

Using (4.2.5) here yields

$$2^{2k+4}AB(1+2^{2k}B\alpha)iw+2^{6k+4}AB^3\beta iw+2^{6k+4}AB^3\gamma i$$
$$= 2^{2k+4}AB(1+2^{2k}B\alpha+2^{4k}B^2\beta)iw+2^{6k+4}AB^3\gamma i \in \mathbb{Z}[\Pi,\overline{\Pi}]\,. \tag{4.2.6}$$

Now use (4.2.3) with (4.2.6) to get

$$2^4A(1+2^{2k}B\alpha+2^{4k}B^2\beta+2^{6k}B^3\gamma)i = 2^4AQi \in \mathbb{Z}[\Pi,\overline{\Pi}]\,. \tag{4.2.7}$$

Finally use (4.2.2) and (4.2.4) with (4.2.7) to get

$$2^{2k+7}ABQiw \in \mathbb{Z}[\Pi,\overline{\Pi}] \quad \text{and} \quad 2^{k+5}A^2Qiw^2 \in \mathbb{Z}[\Pi,\overline{\Pi}]\,.$$

Since $\mathbb{Z}[\Pi,\overline{\Pi}]=\mathbb{Z}[\pi,\overline{\pi}]$, we have

$$2^{2k+3}Bw \in \mathbb{Z}[\pi,\overline{\pi}]\,, \quad 2^{k+1}Aw^2 \in \mathbb{Z}[\pi,\overline{\pi}]\,, \quad 2^4AQi \in \mathbb{Z}[\pi,\overline{\pi}]\,,$$
$$2^{2k+7}ABQiw \in \mathbb{Z}[\pi,\overline{\pi}]\,, \quad 2^{k+5}A^2Qiw^2 \in \mathbb{Z}[\pi,\overline{\pi}]\,.$$

These imply that

$$[\mathbb{Z}[i,w]:\mathbb{Z}[\pi,\overline{\pi}]] \mid 2^{6k+20}A^5B^2Q^3\,.$$

By Lemma 4.2.2 we have $[\mathcal{O}_K:\mathbb{Z}[i,w]] \mid 2^5[\mathcal{O}_{K_0}:\mathbb{Z}[w]]^2$, so finally we get

$$[\mathcal{O}_K:\mathbb{Z}[\pi,\overline{\pi}]] \mid 2^{6k+25}A^5B^2Q^3[\mathcal{O}_{K_0}:\mathbb{Z}[w]]^2\,.$$

$\square$

Lemma 4.2.4. *$\nu_2(C) = 3$ and $(C/8)P \equiv 1 \pmod{2^{k-1}}$.*

Proof. Compute

$$\mathrm{Nm}_{K/\mathbb{Q}(i)}(\pm\Pi - 1) = \pm\,\mathrm{Nm}_{K/\mathbb{Q}(i)}(\Pi \mp 1)$$

$$= \pm\det\begin{pmatrix} a_0 \mp 1 + i & 2^k A\gamma & -2^k\gamma(A\alpha - 2^k Bi) \\ -2^{2k}Bi & a_0 \mp 1 + 2^k A\beta + i & 2^k(A(\gamma - \alpha\beta) + 2^k B\beta i) \\ -2^k A & 2^k(A\alpha - 2^k Bi) & a_0 \mp 1 + 2^k A(\beta - \alpha^2) + (1 + 2^{2k}B\alpha)i \end{pmatrix}$$

which looks really bad until we reduce it modulo 2^{k-1} and find that

$$\mathrm{Nm}_{K/\mathbb{Q}(i)}(\pm\Pi - 1) \equiv \pm(a_0 \mp 1 + i)^3 \equiv \pm(\mp 1 + i)^3 = \pm 2 \pm 2i \pmod{2^{k-1}}$$

where we also used the fact that $a_0 \equiv 0 \pmod{2^{k-1}}$. Now this means that, for some integers a and b

$$\mathrm{Nm}_{K/\mathbb{Q}(i)}(\pm\Pi - 1) = \pm(2 + 2^{k-1}a) \pm (2 + 2^{k-1}b)i\,.$$

Taking the absolute norm of this yields

$$\mathrm{Nm}(\pm\Pi - 1) = (2 + 2^{k-1}a)^2 + (2 + 2^{k-1}b)^2$$

$$= 8\left[1 + 2^{k-1}(a+b) + 2^{2k-5}(a^2 + b^2)\right] = C_s P_s\,,$$

which gives $\nu_2(C_s) = 3$ since P is a large odd prime, and $(C/8)P \equiv 1 \pmod{2^{k-1}}$. □

4.3 Parameter Selection

We start by considering the dependence of p on α, β, A, B and k. It is easy to compute that

$$p = 2^{6k-4}\left(\frac{2B^2}{A}\right)^2 + 2^{4k-1}B^2\left(\alpha^2 - \beta\right) + 2^{2k-2}A^2\left((\alpha^2 - \beta)^2 - 4\alpha\gamma\right) + 1 \tag{4.3.1}$$

in which the first is by far the most significant term for practical parameters. The idea is that p can be made large even if α and β remain small by taking k to be large enough (see Remark 4.2.1).

Remark 4.3.1. Depending on k and the other parameters, in the binary representation of p most of the digits are zeros and the ones are grouped together in roughly four spots. This might be relevant for high-performance arithmetic in $\mathbb{F}_p$ on consequently on the Jacobian.

Lemma 4.3.2. *For reasonable parameters*[4] *the number of bits in p is*

$$6k - 1 - 2\nu_2(A) + 4\lfloor \log_2 B \rfloor\,.$$

□

[4]For reasonable parameters the first term in (4.3.1) is by far the largest.

Lemma 4.3.3. *For reasonable parameters, as in Lemma 4.3.2, the number of bits in* $\mathrm{Nm}(\pi - 1)$ *is*

$$18k - 5 - 6\nu_2(A) + 12\lfloor \log_2 B \rfloor .$$

Proof. This follows immediately from the fact that $\mathrm{Nm}(\pi - 1) = p^3 + O(p^{5/2})$. □

How big does it make sense to allow the cofactor C in $\mathrm{Nm}(\pi - 1)$ to be? We already saw in Lemma 4.2.4 that C is necessarily divisible by 8. Recall that Jacobians of hyperelliptic genus 3 curves admit a double large prime index calculus with complexity $\widetilde{O}(p^{4/3})$ (see [GTTD]). This means that Pollard rho operating in the large prime order subgroup yields no further reduction in security as long as

$$p^{4/3} \ll \left(\frac{p^3}{C}\right)^{1/2} ,$$

which simplifies to $C \ll p^{1/3}$. By Lemma 4.3.2 this means that the number of bits in C should be less than $2k$. To obtain practical sizes we need $k \geq 10$, so this is a very large bound.

From Lemma 4.3.2 we see that to have a prime p of N bits we should take k to be at most $\lfloor (N+1)/6 \rfloor$. The smaller k we choose the bigger B can be, which gives us more options for the input to Algorithm 4.2.1. For security reasons it might preferable to have k be as big as possible, as we will see in Section 4.4.

After N (the bit-length of p) and k have been chosen, from the formula in Lemma 4.3.2 we get bounds for B in terms of A:

$$2^{-1/4}(2A)^{1/2}2^{N/4-3k/2} \leq B < (2A)^{1/2}2^{N/4-3k/2} . \tag{4.3.2}$$

If 2^{2m} divides both A^2 and B, then in fact we get a setup identical to something we would get by replacing $k \mapsto k + m$. To avoid such overcounting we only choose B such that $\gcd(A^2, B) \leq 2$. When $A = 1$, any of the integers in the range (4.3.3) is a possible choice for B, but when $A \geq 2$ the divisibility condition $A^2 \mid 2B$ must be satisfied as well as the non-trivial $\gcd(A^2, B) \leq 2$. But together these can only hold when $A \leq 2$ so the only possible values for A are 1 and 2.

In the case $A = 1$, (4.3.2) becomes

$$2^{-1/4}2^{N/4-3k/2+1/2} \leq B < 2^{N/4-3k/2+1/2} . \tag{4.3.3}$$

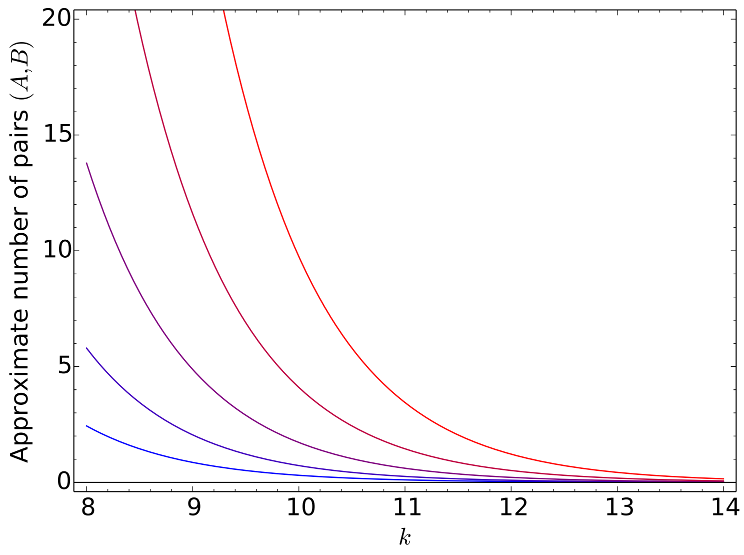
In the case $A = 2$, (4.3.2) becomes

$$2^{-1/4}2^{N/4-3k/2+1} \leq B < 2^{N/4-3k/2+1} , \tag{4.3.4}$$

where in addition we require $B \equiv 2 \pmod 4$ to ensure $\gcd(A^2, B) \leq 2$. Hence, the number of possible Bs is very closely approximated by

$$\begin{cases} \left(1 - 2^{-1/4}\right) 2^{N/4-3k/2+1/2} & \text{when } A = 1; \\ \left(1 - 2^{-1/4}\right) 2^{N/4-3k/2-1} & \text{when } A = 2. \end{cases} \tag{4.3.5}$$

Figure 4.1: Approximate number of pairs (A, B) available as a function of k, for $N = 60, 65, 70, 75, 80$



From (4.3.5) we get that the approximate number of pairs (A, B) that yield primes p of N bits with a fixed k is very closely approximated by

$$\left(1 - 2^{-1/4}\right)\left(2^{1/2} + 2^{-1}\right) 2^{N/4-3k/2} . \tag{4.3.6}$$

It is worth pointing out that for a fixed N this number decreases exponentially as a function of k. In Figure 4.1 we have plotted (4.3.6) as a function of k for some interesting values of N.

Next we consider what restrictions there are on k. When $A = 1$, to find any values for B we certainly need $2^{N/4-3k/2+1/2} > 1$, so $N/4 - 3k/2 + 1/2 > 0$. When $A = 2$, we need $2^{N/4-3k/2} > 1$. So to find any pairs (A, B) resulting in a prime of the right size, we need k to satisfy

$$\begin{cases} k \le \lceil N/6 + 1/3 \rceil - 1 & \text{when } A = 1; \\ k \le \lceil N/6 \rceil - 1 & \text{when } A = 2. \end{cases}$$

Note that some values of N are particularly bad in the sense that only $A = 1$ might yield pairs (A, B) for the largest possible k, namely those for which $\lceil N/6 + 1/3 \rceil = \lceil N/6 \rceil + 1$.

Based on these observations, we suggest the following approach for choosing suitable parameters:

1) Choose a target bit length N for p.

2) Choose a minimum value $k_{\min}$ for k.

3) List all integer triples $(A = 1, B, k)$ where

$$k_{\min} \leq k \leq \lceil N/6 + 1/3 \rceil - 1\,, \quad 2^{N/4-3k/2+1/4} \leq B < 2^{N/4-3k/2+1/2}\,,$$

and all integer triples $(A = 2, B, k)$ where $B \equiv 2 \pmod 4$ and

$$k_{\min} \leq k \leq \lceil N/6 \rceil - 1\,, \quad 2^{N/4-3k/2+3/4} \leq B < 2^{N/4-3k/2+1}\,.$$

4) Choose bounds for α (e.g. $[-30, 30]$) and bounds for β (e.g. $[-1000, 0]$)[5]. Choose some reasonable bound $\Delta_{\max}$ for the discriminant Δ (e.g. 10^5).

5) Loop over triples (A, B, k) and over all (α, β) in the chosen ranges, and for each of these compute $\gamma = \alpha\beta - 2B/A^2$ and Δ. Discard those where $\Delta > \Delta_{\max}$.

6) Run Algorithm 4.2.1 for all tuples $(\alpha, \beta, \gamma, A, B, k)$ until you find a set of parameters that satisfies the primality conditions and the class number conditions.

We present this in algorithm form in Algorithms 4.3.1, 4.3.2. First Algorithm 4.3.1 constructs a set of potentially good curve parameters that are then given to Algorithm 4.3.2, which tests them against several conditions and if all are met outputs sextic CM fields K/K_0 together with ordinary p-Weil numbers $\pi \in \mathcal{O}_K$, such that $\pi\overline{\pi} = p$ is an N-bit prime and all the conditions required by the genus 3 CM method are met as given in Section 4.2.

If either Algorithm 4.3.1 or Algorithm 4.3.2 returns an empty list, a smaller $k_{\min}$ should be chosen. According to (4.3.6) and Figure 4.1 allowing k to be just slightly smaller is likely to significantly increase the probability of getting a non-empty output. Alternatively a larger $\Delta_{\max}$, a larger $\alpha_{\max}$ and a smaller $\beta_{\min}$ may be used, but larger discriminants Δ are much more likely to produce fields K with large class numbers that make the genus 3 CM method inconvenient or impossible to do in practice.

Remark 4.3.4. Computing class numbers is not a problem. Assuming the *Generalized Riemann Hypothesis (GRH)* the complexity of computing the class number of a number field with discriminant Δ is

$$L_\Delta(1/2, \sqrt{2} + o(1)) := \exp\left((\sqrt{2} + o(1)) \cdot (\log \Delta)^{1/2} (\log\log \Delta)^{1/2} \right)$$

according to [Bac]. Since the discriminants are assumed to be fairly small, the class number is expected to be reasonably small for the computation to be fast even if the GRH is not assumed and Minkowski's bound is used. Such computations are done in a few seconds using either SAGE or Magma [BCP] and do not pose any problem.

A few more comments are in order:

[5]Having β be negative and in general larger in absolute value than α ensures that the discriminant Δ is positive.

Algorithm 4.3.1 Construction of curve parameters

```
1: Input: $\alpha_{\max}$, $\beta_{\min}$, $\Delta_{\max}$, $k_{\min}$
2: for $k \in [k_{\min}, \lceil N/6 + 1/3 \rceil - 1]$ do
3:     Let $B_{\min} := \lceil 2^{N/4-3k/2+1/4} \rceil$ and $B_{\max} := \lceil 2^{N/4-3k/2+1/2} \rceil - 1$.
4:     for $B \in [B_{\min}, B_{\max}]$ do
5:         for $\alpha \in [-\alpha_{\max}, \alpha_{\max}]$ and $\beta \in [\beta_{\min}, 0]$ do
6:             Let $\gamma := \alpha\beta - 2B/A^2$.
7:             Let $\Delta := \alpha^2\beta^2 - 4\beta^3 - 4\alpha^3\gamma - 27\gamma^2 + 18\alpha\beta\gamma$.
8:             if $0 < \Delta \leq \Delta_{\max}$ then
9:                 Record the tuple $(\alpha, \beta, \gamma, A = 1, B, k)$.
10:            end if
11:        end for
12:    end for
13: end for
14: for $k \in [k_{\min}, \lceil N/6 \rceil - 1]$ do
15:     Let $B_{\min} := \lceil 2^{N/4-3k/2+3/4} \rceil$ and $B_{\max} := \lceil 2^{N/4-3k/2+1} \rceil - 1$.
16:     for $B \in [B_{\min}, B_{\max}]$ such that $B \equiv 2 \pmod 4$ do
17:         for $\alpha \in [-\alpha_{\max}, \alpha_{\max}]$ and $\beta \in [\beta_{\min}, 0]$ do
18:             Let $\gamma := \alpha\beta - 2B/A^2$.
19:             Let $\Delta := \alpha^2\beta^2 - 4\beta^3 - 4\alpha^3\gamma - 27\gamma^2 + 18\alpha\beta\gamma$.
20:             if $0 < \Delta \leq \Delta_{\max}$ then
21:                 Record the tuple $(\alpha, \beta, \gamma, A = 2, B, k)$.
22:             end if
23:         end for
24:     end for
25: end for
26: Output: List of tuples $(\alpha, \beta, \gamma, A, B, k)$.
```

1) The bound $k_{\min}$ should not be too small. The larger it is, or really the larger k is, the more work the attacker must do to perform an isogeny attack as will be explained and discussed in Section 4.4.

2) The symmetric choice for the α interval and the choice for the β interval are to guarantee that the discriminant Δ is positive.

3) The index $[\mathcal{O}_{K_0} : \mathbb{Z}[w]]$ is very likely to be small. It would be preferable to have it be 1 to simplify the CM method, but also larger indices are possible to deal with as is explained in [Wen].

4) The polynomial $f(x)$ is almost certainly irreducible.

5) The bound $p^{4/9}$ for the size of the prime divisors of Q is explained in Section 4.4.

Algorithm 4.3.2 Construction of CM data

```
1: **Input:** A list of tuples $(\alpha, \beta, \gamma, A, B, k)$ constructed by Algorithm 4.3.1.
2: **for all** input tuples $(\alpha, \beta, \gamma, A, B, k)$ **do**
3:     Let $a_0 := 2^{3k-2}(2B^2/A) + 2^{k-1}A(\alpha^2 - \beta)$.
4:     Let $p := a_0^2 + 2^{2k}A^2\alpha\gamma + 1$.
5:     **if** $p$ is not prime **then**
6:         **Continue**
7:     **end if**
8:     Let $Q := 1 + 2^{2k}B\alpha + 2^{4k}B^2\beta + 2^{6k}B^3\gamma$.
9:     **if** $Q$ has prime divisors smaller than $p^{4/9}$ **then**
10:        **Continue**
11:    **end if**
12:    Let $f(x) := x^3 + \alpha x^2 + \beta x + \gamma$.
13:    **if** $f$ is not irreducible over $\mathbb{Q}$ **then**
14:        **Continue**
15:    **end if**
16:    Let $K_0$ be the field generated by a root $w$ of $f(x)$.
17:    **if** $[\mathcal{O}_{K_0} : \mathbb{Z}[w]]$ is not small **then**
18:        **Continue**
19:    **end if**
20:    Let $K := K_0(i)$.
21:    **if** $p$ is ramified in $K$ **then**
22:        **Continue**
23:    **end if**
24:    Let $\Pi := a_0 - 2^k Aw^2 + i(1 - 2^{2k}Bw)$.
25:    Compute $N_\pm := \mathrm{Nm}\,(\pm\Pi - 1)$.
26:    **if** $N_s$ equals a prime times a cofactor $C_s$ ($C_s$ at most $k$ bits) for a sign $s$ **then**
27:        Let $P := N_s/C_s$, $C := C_s$ and $\pi := s\Pi$.
28:        **if** $|\varphi(\pi + \overline{\pi})| > 2p^{1/2}$ for some embedding $\varphi : \mathcal{O}_{K_0} \hookrightarrow \mathbb{R}$ **then**
29:            **Continue**
30:        **end if**
31:    **else**
32:        **Continue**
33:    **end if**
34:    **if** the class number of $K_0$ is not 1 **then**
35:        **Continue**
36:    **end if**
37:    Compute the class number $\#\,\mathrm{Cl}(K)$ of $K$.
38:    Record the tuple $(\alpha, \beta, \gamma, A, B, k, a_0, p, Q, K_0, K, P, C, \pi, \#\,\mathrm{Cl}(K))$.
39: **end for**
40: **Output:** List of tuples $(\alpha, \beta, \gamma, A, B, k, a_0, p, Q, K_0, K, P, C, \pi, \#\,\mathrm{Cl}(K))$.
```

6) An explicit computation reveals that the inequality $|\varphi(\pi+\overline{\pi})| \leq 2p^{1/2}$ should hold for all real embeddings $\varphi : \mathcal{O}_{K_0} \hookrightarrow \mathbb{R}$ as long as k is of practically relevant size.

4.4 Special Properties of the Isogeny Class

We start by presenting a result from [Mil2] explaining the relation between the automorphisms of Jac_C and C.

Theorem 4.4.1 ([Mil2])**.** *If $\mathcal{A}$ is the Jacobian variety of a curve C with canonical principal polarization λ, then*

$$\operatorname{Aut}\mathcal{C} \cong \begin{cases} \operatorname{Aut}(\mathcal{A},\lambda) & \text{if } C \text{ is hyperelliptic;} \\ \operatorname{Aut}(\mathcal{A},\lambda)/\{\pm 1\} & \text{if } C \text{ is non-hyperelliptic.} \end{cases}$$

In [Wen] Weng used Theorem 4.4.1 to find Jacobians of genus 3 hyperelliptic curves (rather than non-hyperelliptic) as follows.

Theorem 4.4.2. *If $\mathcal{A}$ is a simple Jacobian variety of a curve C of genus $g \geq 2$ and $i \in \operatorname{End}\mathcal{A}$, then C must be hyperelliptic.*

Proof. We know that 4 divides $|\operatorname{Aut}(\mathcal{A},\lambda)|$ so by Theorem 4.4.1 we see that C must have an automorphism φ of order at least 2. But then $C/\langle\varphi\rangle$ sits inside C and the Jacobian Jac_C is not simple unless $C/\langle\varphi\rangle$ has genus 0. Hence there is a degree 2 cover $C \to \mathbb{P}^1$ so C is hyperelliptic. □

Suppose from now on that a CM field K/K_0 and a p-Weil number π have been constructed using Algorithms 4.3.1,4.3.2. Suppose the genus 3 CM method of Weng has been used to construct an abelian variety $\mathcal{A}/\mathbb{F}_p$, which is the Jacobian of a hyperelliptic curve $C/\mathbb{F}_p$ of genus 3 such that $\operatorname{End}\mathcal{A} = \mathcal{O}_K$ (recall Section 2.6).

Lemma 4.4.3. *Let ℓ be a prime and $\mathcal{A} \to \mathcal{A}/\mathcal{K} = \mathcal{B}$ an isogeny to a 3-dimensional abelian variety $\mathcal{B}/\mathbb{F}_p$, such that $\operatorname{End}\mathcal{B} = \mathcal{O} \neq \mathcal{O}_K$. Then $\mathcal{O}$ is an order in $\mathcal{O}_K$ containing $\mathbb{Z}[\pi,\overline{\pi}]$. If the kernel $\mathcal{K}$ is contained in $\mathcal{A}[\ell^n]$ for some prime ℓ, then $\ell \mid 2BQ[\mathcal{O}_{K_0} : \mathbb{Z}[w]]$.*

Proof. Recall from Section 2.5 that the endomorphism ring of $\mathcal{B}$ is an order $\mathcal{O} \subseteq \mathcal{O}_K$ containing $\mathbb{Z}[\pi,\overline{\pi}]$.

For any endomorphism $\varphi \in \mathcal{O}_K$ of $\mathcal{A}$ we have a sequence of abelian varieties

$$\mathcal{B} = \mathcal{A}/\mathcal{K} \longrightarrow \mathcal{A}/\mathcal{A}[\ell^n] \xrightarrow{[\ell^n]} \mathcal{A} \xrightarrow{\varphi} \mathcal{A} \longrightarrow \mathcal{A}/\mathcal{K} = \mathcal{B}$$

which shows that $\ell^n\varphi \in \mathcal{O}$, and consequently $\ell^n\mathcal{O}_K \subseteq \mathcal{O}$. Since $\mathcal{O}$ is a ring, we in fact have

$$\mathbb{Z} + \ell^n\mathcal{O}_K \subseteq \mathcal{O}\,.$$

Since $[\mathcal{O}_K : \mathbb{Z} + \ell^n \mathcal{O}_K] = \ell^{5n}$, we have a diagram of orders (sublattices)

$$\begin{tikzcd}
 & \mathcal{O}_K & \\
 & \mathcal{O} \arrow[u, "\neq 1"'] & \\
 & \mathbb{Z}[\pi,\overline{\pi}] + \ell^n \mathcal{O}_K \arrow[u] & \\
\mathbb{Z}[\pi,\overline{\pi}] \arrow[ur] \arrow[uuur, bend left, "2^{6k+25}A^5B^2Q^3[\mathcal{O}_{K_0}:\mathbb{Z}[w]]^2"] & & \mathbb{Z} + \ell^n \mathcal{O}_K \arrow[ul] \arrow[uuul, bend right, "\ell^{5n}"']
\end{tikzcd}$$

where the labels denote numbers that the relative indices are divisible by. From this it is clear that ℓ must divide $2BQ[\mathcal{O}_{K_0} : \mathbb{Z}[w]]$ (since A is either 1 or 2). □

Theorem 4.4.4. *Let $\mathcal{K} \subset \mathcal{A}[2^n N]$ where N is divisible by powers of odd prime divisors of $B[\mathcal{O}_{K_0} : \mathbb{Z}[w]]$, and suppose $\mathcal{K}$ is stable under π. Suppose $\mathcal{B} = \mathcal{A}/\mathcal{K}$ is the Jacobian of a non-hyperelliptic curve defined over $\mathbb{F}_p$.*

a) $\mathcal{A}[2^n N] \cong \mathcal{A}[2^n] \oplus \mathcal{A}[N]$ and under this decomposition $\mathcal{K}$ decomposes as $\mathcal{K} \cong \mathcal{K}[2^n] \oplus \mathcal{K}[N]$ into subgroups of the two summands. Both $\mathcal{K}[2^n]$ and $\mathcal{K}[N]$ are π-stable.

b) $\mathcal{K}$, $\mathcal{K}[2^n]$ and $\mathcal{K}[N]$ are stable under $\overline{\pi} = p/\pi$.

c) $\mathcal{K}$ must contain a point of power-of-2 order at least $2^{k+\nu_2(A)+1}$.

d) If $\mathcal{B}$ has real multiplication (RM) by $2^k A w^2$, then $\mathcal{K}$ must contain a point of power-of-2 order at least $2^{2k+\nu_2(B)+1}$.

e) $\mathcal{B}$ can not have RM by $2^{\lfloor (k+\nu_2(A))/2 \rfloor} w$.

Proof. Any abelian group of exponent NM, where N and M are relatively prime, is isomorphic to a direct sum of its N-torsion and M-torsion subgroups. This follows directly from the structure theorem of finitely generated abelian groups. Hence $\mathcal{A}[2^n N] \cong \mathcal{A}[2^n] \oplus \mathcal{A}[N]$ and $\mathcal{K} \cong \mathcal{K}[2^n] \oplus \mathcal{K}[N]$. Since π commutes with $\mathbb{Z}$,

$$2^n \pi \mathcal{K}[2^n] = \pi\, 2^n \mathcal{K}[2^n] = 0\,, \quad \text{and} \quad N\pi\mathcal{K}[N] = \pi\, N\mathcal{K}[N] = 0\,,$$

so both subgroups of $\mathcal{K}$ are stable under π. This proves a).

In fact, it is true that $\pi\mathcal{K} = \mathcal{K}$. First, note that π satisfies a degree 6 minimal polynomial $\pi g(\pi) + p^3 = 0$, where $g(x)$ is some polynomial in $\mathbb{Z}[x]$ of degree 5. Next, $p^3\mathcal{K} = \mathcal{K}$ because $\gcd(p, 2^n N) = 1$, meaning we can find integers a and b such that $a\,p^3 + b\,2^n N = 1$, and write

$$p^3\mathcal{K} \subseteq \mathcal{K} = (a\,p^3 + b\,2^n N)\mathcal{K} = a\,p^3\mathcal{K} \subseteq p^3\mathcal{K}\,.$$

Similarly of course $p\mathcal{K} = \mathcal{K}$. We get

$$\pi\mathcal{K} \subseteq \mathcal{K} = -p^3\mathcal{K} = \pi g(\pi)\mathcal{K} \subseteq \pi\mathcal{K}$$

since $g(\pi)\mathcal{K} \subseteq \mathcal{K}$, and so

$$\overline{\pi}\mathcal{K} = \overline{\pi}\pi\mathcal{K} = p\mathcal{K} = \mathcal{K}\,.$$

Again, as in the proof of a), $\overline{\pi}$ must leave the two subgroups $\mathcal{K}[2^n]$ and $\mathcal{K}[N]$ invariant. This proves b).

By definition of π,

$$i = \pm\pi - a_0 + 2^kAw^2 + 2^{2k}Biw\,. \tag{4.4.1}$$

Since $\mathcal{B}$ is defined over $\mathbb{F}_p$, the kernel $\mathcal{K}$ must be stable under π and by Theorem 4.4.2 it can not be stable under i. By (4.4.1) the summand $\mathcal{K}[2^n]$ is stable under i unless it contains points of order at least $2^{k+\nu_2(A)+1}$. Note that $\gcd\left(B[\mathcal{O}_{K_0} : \mathbb{Z}[w]], Q\right) = 1$ since Q is assumed to have only large factors. Therefore, $\mathcal{K}[N] = 2^7AQ\mathcal{K}[N]$, and so

$$\begin{aligned} i\mathcal{K}[N] &= i(2^7AQ\mathcal{K}[N]) \\ &\subseteq 2^7AQ(\pm\pi - a_0)\mathcal{K}[N] + 2^{k+7}A^2Qw^2\mathcal{K}[N] + 2^{2k+7}A^2BQiw\mathcal{K}[N] \end{aligned}$$

where $2^{k+7}A^2Qw^2 \in \mathbb{Z}[\pi,\overline{\pi}]$ and $2^{2k+7}A^2BQiw \in \mathbb{Z}[\pi,\overline{\pi}]$ by Theorem 4.2.3. Since $\mathbb{Z}[\pi,\overline{\pi}]$ leaves $\mathcal{K}$ invariant, we conclude that the only way to get to a non-hyperelliptic Jacobian $\mathcal{A}/\mathcal{K}$ is by taking $\mathcal{K}$ to have points of power-of-2 order at least $2^{k+\nu_2(A)+1}$. We are done with c).

For d), suppose that $\mathcal{B}$ has RM by 2^kAw^2, meaning 2^kAw^2 leaves $\mathcal{K}$ invariant. From (4.4.1) we see, just like in the proof of c), that $\mathcal{K}$ must contain points of power-of-2 order at least $2^{2k+\nu_2(B)+1}$.

To prove e), consider first what happens if $\mathcal{B}$ has RM by 2^kAw^2. As in the proof of c), $\mathcal{K} = Q\mathcal{K}$. Using this and (4.4.1), we can write

$$\begin{aligned} i\mathcal{K} &\subseteq (\pm\pi - a_0)\mathcal{K} + 2^kAw^2\mathcal{K} + 2^{2k}Biw\mathcal{K} \\ &= (\pm\pi - a_0)\mathcal{K} + 2^kAw^2\mathcal{K} + 2^{2k-\nu_2(A)-4}Bw(2^4AQi)\mathcal{K}\,. \end{aligned}$$

But $(\pm\pi - a_0)\mathcal{K} \subseteq \mathcal{K}$ and we assumed $2^kAw^2\mathcal{K} \subseteq \mathcal{K}$, and by Theorem 4.2.3 $2^4AQi \in \mathbb{Z}[\pi,\overline{\pi}]$, so in fact

$$i\mathcal{K} \subseteq \mathcal{K} + 2^{2k-\nu_2(A)-4}Bw\mathcal{K}\,. \tag{4.4.2}$$

Therefore, if $\mathcal{B}$ has RM by $2^{2k-\nu_2(A)-4}Bw$ it also has CM by i, which can not be the case by Theorem 4.4.2. We conclude that $\mathcal{B}$ can not have RM by $2^{2k-\nu_2(A)-4}Bw$ if it has RM by 2^kAw^2.

Now, if $\mathcal{B}$ does have RM by $2^{\lfloor(k+\nu_2(A))/2\rfloor}w$, then it also has RM by 2^kAw^2 and, according the above argument, not by $2^{2k-\nu_2(A)-4}Bw$. It is very easy to see that

$$\lfloor(k+\nu_2(A))/2\rfloor < 2k - \nu_2(A) + \nu_2(B) - 4\,,$$

so $\mathcal{B}$ having RM by $2^{\lfloor(k+\nu_2(A))/2\rfloor}w$ means that it also has RM by $2^{2k-\nu_2(A)-4}Bw$. This is a contradiction, proving that $\mathcal{B}$ can not have RM by $2^{\lfloor(k+\nu_2(A))/2\rfloor}w$. □

According to Theorem 4.4.4 the attacker must necessarily compute an isogeny whose kernel involves high power-of-2 torsion points. We next argue that it is not feasible to do this in one step. Lemma 4.2.4 shows that $\#\mathcal{A}(\mathbb{F}_p)[2] = 8$, and the following result shows that the remaining 2-torsion points are defined over $\mathbb{F}_{p^2}$.

Lemma 4.4.5. *The subgroup $\mathcal{A}[1+i]$ is a 3-dimensional subspace of $\mathcal{A}[2]$. All points of $\mathcal{A}[1+i]$ are defined over $\mathbb{F}_p$ and the rest of the points in $\mathcal{A}[2]$ are defined over $\mathbb{F}_{p^2}$.*

Proof. Recall that $i \in \operatorname{End}\mathcal{A}$ and that

$$\pm\,\pi = a_0 - 2^k\,Aw^2 + i(1 - 2^{2k}\,Bw)\,. \tag{4.4.3}$$

The subgroup $\mathcal{A}[1+i]$ consists of points $x \in \mathcal{A}$ such that $(1+i)x = 0$. But for such a point $(1-i)(1+i)x = 2x = 0$, so $x \in \mathcal{A}[2]$. Hence $\mathcal{A}[1+i]$ is a subgroup of $\mathcal{A}[2]$. Moreover, by (4.4.3) (recall that a_0 is even) we have $\pi x = ix = x$, so $\mathcal{A}[1+i] \subseteq \mathcal{A}(\mathbb{F}_p)[2]$. Likewise, $\mathcal{A}(\mathbb{F}_p)[2] \subseteq \mathcal{A}[1+i]$, so in fact $\mathcal{A}[1+i] = \mathcal{A}(\mathbb{F}_p)[2]$. Hence $\mathcal{A}[1+i]$ is a 3-dimensional $\mathbb{F}_p$-rational subspace of $\mathcal{A}[2]$.

According to (4.4.3) π acts as i on $\mathcal{A}[2]$ and so π^2 acts as 1, which means that the rest of the points in $\mathcal{A}[2]$ are defined over $\mathbb{F}_{p^2}$. □

Lemma 4.4.6. *The smallest extension of $\mathbb{F}_p$ over which a point of order 2^M of $\mathcal{A}$ is guaranteed to be found is of degree 2^{M-1}. This is also the smallest extension where such a point can be "expected" to be found.*

Proof. It follows from the endomorphism $i \in \operatorname{End}\mathcal{A}$ of order 4 that $\mathcal{A}[2]$ has at least one point that is stable under π. This can now be lifted to $\mathcal{A}[4]$ using Cantor's hyperelliptic division polynomials [Can] and can be repeated until a point of order 2^M is found[6]. Performing the lifting requires extraction of square-roots, which will result in elements in a quadratic extension field.

We found in Lemma 4.4.5 that $\mathcal{A}(\mathbb{F}_p)[2] = \mathcal{A}[1+i]$. It follows that the smallest field extension of $\mathbb{F}_p$ where one can expect (in the sense that one must almost certainly go to a quadratic extension whenever a division by 2 is performed) to find a point of order 2^M, has degree 2^{M-1}. □

Corollary 4.4.7. *If $\mathcal{A} \to \mathcal{A}/\mathcal{K} = \mathcal{B}$ is an isogeny where the target $\mathcal{B}$ is the Jacobian of a non-hyperelliptic curve, then one should expect the points of $\mathcal{K}$ to have coordinates in an extension of $\mathbb{F}_p$ of degree at least $2^{k+\nu_2(A)}$.*

Furthermore, if $\mathcal{B}$ has RM by 2^kAw^2, then one should expect the points of $\mathcal{K}$ to have coordinates in an extension of $\mathbb{F}_p$ of degree at least $2^{2k+\nu_2(B)}$.

Proof. Follows immediately from Theorem 4.4.4 and Lemma 4.4.6. □

[6]This lifting process is studied and demonstrated in the case of genus 2 in [GS].

4.5 Explicitly Computable Isogenies

By an *explicitly computable isogeny* we mean that given the theta null point of the source principally polarized abelian variety and the theta coordinates of points in a finite subgroup, there is an efficient and practical algorithm to compute the theta null point of the quotient principally polarized abelian variety and find theta coordinates of the image of any $\mathbb{F}_p$-valued point. We also need to be able to recover the Jacobian structures of the source and the target abelian varieties and be able to express all points as divisors on the corresponding curves.

Computing an isogeny to a non-hyperelliptic Jacobian in one step from $\mathcal{A}$ is unlikely to be possible due to Theorem 4.4.4 and Corollary 4.4.7. The attacker would need to find a π-stable kernel containing points of very high power-of-2 torsion, which would only be defined over large extension fields. Even worse, the algorithms of [LR, CR, Rob] require computing theta coordinates for the points in the kernel, which typically live in even larger field extensions. So instead of doing to computation in one step the attacker is practically forced to compute a chain of isogenies between Jacobians of hyperelliptic curves over $\mathbb{F}_p$, until the Jacobian of a non-hyperelliptic curve is encountered. The smallest possible length of the chain clearly depends directly on k.

Maximal Isotropic Isogenies

Currently (ℓ, ℓ, ℓ)-isogenies between Jacobian varieties are by far the easiest isogenies to compute[7]. The reason is that such an isogeny $f : \mathcal{A} \to \mathcal{B}$ respects principal polarizations in the sense that for a principal polarization on $\mathcal{A}$ determined by a degree 1 line sheaf $\mathcal{L}$ there is always an induced polarization $\mathcal{M}$ on the target $\mathcal{B}$ such that $f^*\mathcal{M} \cong \mathcal{L}^{\deg f}$. For instance, this means that the target is guaranteed to be principally polarized and hence isomorphic to the Jacobian of a genus 3 curve according to Theorem 2.6.7, although as we pointed out in Section 2.6 and again in Section 4.1 the target abelian variety might only be isomorphic to the Jacobian over a quadratic extension of the base field if the target curve is non-hyperelliptic, in which case it would not be useful for the attacker. These techniques were developed in references [LR, CR, Rob] and, while only implemented in genus 2, should just as well work to compute (ℓ, ℓ, ℓ)-isogenies between Jacobians of genus 3 curves.

Much earlier Smith [Smi] computed certain $(2, 2, 2)$-isogenies between Jacobians of genus 3 curves and demonstrated the isogeny attack with practical examples. While his method is much simpler and faster than the heavy machinery of [LR, CR, Rob], it only works for certain maximal isotropic *tractable* subgroups $\mathcal{K} \cong (\mathbb{Z}/2\mathbb{Z})^3$ of $\mathcal{A}[2]$, whose existence depends on the form of the hyperelliptic polynomial. In particular, Smith found that only 18.57% of hyperelliptic genus 3 curves are vulnerable.

When ℓ is odd, the complexity of computing an (ℓ, ℓ, ℓ)-isogeny is at least $\widetilde{O}(\ell^3)$ operations in the field of definition of the points of $\mathcal{K}$ using the methods of [CR]. In Algorithm 4.3.2 we restricted the prime factors of Q to be at least $p^{4/9}$ to ensure that computing an isogeny by a

[7]At least when the dimension of the varieties is greater than 1. In the case of elliptic curves computing isogenies is significantly easier (see [Sil1, Was]).

factor of Q is no easier than breaking the DLP on the hyperelliptic Jacobian using the $\widetilde{O}(p^{4/3})$ index calculus. In reality it would be extremely difficult to compute isogenies by such large kernels, even much more difficult than suggested by the complexity estimate, because one would have to deal with extremely large extension fields of $\mathbb{F}_p$ to describe the points in the kernel. This means that the attacker faces exactly the situation of Theorem 4.4.4, implying that the only isogenies the attacker can directly profit from are either (ℓ, ℓ, ℓ)-isogenies or cyclic ℓ-isogenies, where ℓ is a power of 2.

Cyclic Isogenies

Very recently there has been some progress in computing *horizontal isogenies*[8] with cyclic kernels in the case of genus 2 [Dud], assuming both the source and the target Jacobians have real multiplication (RM) by the maximal order $\mathcal{O}_{K_0}$, or at least by some "large enough" order, but this seems to be hard to make precise. It is not clear at all if the same methods can be extended to work for genus 3 curves. However, the requirements on the ring of real multiplication are so central to the algorithm that it seems very unlikely it could be made to work unless both the source and the target varieties have RM by (almost) all of $\mathcal{O}_{K_0}$. Now, Theorem 4.4.4 implies that the ring of real multiplication on $\mathcal{B}$ is smaller than on $\mathcal{A}$. Part d) implies that unless the kernel contains very high power-of-2 torsion points, $2^k A w^2$ does not act on $\mathcal{B}$. Note that by Theorem 4.2.3 $2^{k+1} A w^2 \in \mathbb{Z}[\pi, \overline{\pi}]$, so in some sense $\mathcal{B}$ is as far as possible from having w^2 act on it. By Theorem 4.2.3 we know that $2^{2k+3} B w \in \mathbb{Z}[\pi, \overline{\pi}]$ so it acts on $\mathcal{B}$, but by part e) of $2^{\lfloor (k+\nu_2(A))/2 \rfloor} w$ does not. This gives a strong reason to believe that computing isogenies with cyclic kernel is unlikely to help the attacker reach a non-hyperelliptic Jacobian unless remarkable theoretical progress is made in explicit isogeny algorithms.

Chains of Isogenies

For the attacker the best course of action seems to be to compute a chain of $(2, 2, 2)$-isogenies, or possibly a chain consisting of $(2^m, 2^m, 2^m)$-isogenies for small integers m, in such a way that each Jacobian in the chain is defined over $\mathbb{F}_p$. This means that each relative kernel must be invariant under π. Due to the nature of the explicit isogeny algorithms at each step it is necessary to identify the target as the Jacobian of a (hyperelliptic) curve, map the DLP to this particular Jacobian and proceed with finding the next kernel in the chain. By Theorem 4.4.4 the kernel of the composition of the isogenies in the chain should contain points of power-of-2 order at least $2^{k+\nu_2(A)+1}$, which means that the attacker would need to compute e.g. at least $k + \nu_2(A) + 1$ $(2, 2, 2)$-isogenies. It is not clear how feasible such a chain of isogenies would be to find and to compute, and a detailed analysis is left for future work.

[8]Horizontal isogenies are isogenies that do not change the endomorphism ring.

Even though our solution to the threat of isogeny attacks is not complete in the sense that there is still a possible path left open for the attacker, we have presented several novel techniques for preventing most isogeny attacks from working.

Of course one should keep in mind that according to the results of Chapter 3 isogeny attacks are not a threat when p is large enough, say when $p > 2^{80}$, because of the massive memory requirements of the index calculus attack. Instead, the question of preventing isogeny attacks is only relevant in the lower part of the range $2^{60} < p < 2^{80}$. Such relatively "low security" cryptosystems using extremely small fields certainly have their uses, but are currently not practical due to the persisting threat of isogeny attacks and the lack of efficient enough arithmetic.

Bibliography

[ADH] L. M. Adleman, J. DeMarrais, M.-D. Huang, *A subexponential algorithm for discrete logarithms over hyperelliptic curves of large genus over GF(q)*, Theoret. Comput. Sci. **226** (1999), no. 1-2, pp. 7–18.

[Adl] L. Adleman, *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, In 20th Annual Symposium on Foundations of Computer Science, IEEE (1979), 55–60.

[AM] A. Atkin, F. Morain, *Elliptic curves and primality proving*, Mathematics of computation **61**, no. 203 (1993), 29–68.

[Bac] E. Bach, Eric, *Explicit bounds for primality testing and related problems*, Mathematics of Computation **55**, no. 191 (1990), 55–380.

[BCLS] D. J. Bernstein, C. Chuengsatiansup, T. Lange, P. Schwabe, *Kummer strikes back: new DH speed records*, In Advances in Cryptology–ASIACRYPT 2014, pp. 317-337, Springer Berlin Heidelberg, 2014.

[BCHL] J. W. Bos, C. Costello, H. Hisil, K. Lauter, *Fast Cryptography in Genus* 2, Advances in Cryptology - EUROCRYPT 2013, Lecture Notes in Comput. Sci. **7881**, Springer Berlin (2013), 194–210.

[BCP] W. Bosma, J. Cannon, C. Playoust, *The Magma algebra system I: The user language*, Journal of Symbolic Computation **24**, no. 3 (1997), 235–265.

[Can] D. G. Cantor, *On the analogue of the division polynomials for hyperelliptic curves*, Journal fur die reine und angewandte Mathematik, **447** (1994), 91–146.

[Can2] D. G. Cantor, *Computing in the Jacobian of a hyperelliptic curve*, Mathematics of computation **48**, no. 177 (1987), 95–101.

[Che] *Formulas for the solutions of quadratic equations over GF*(2^m), IEEE Trans. Inform. Theory **28** (1982), no. 5, 792—794.

[CM] H. Cohen, J. Martinet, *Class groups of number fields: numerical heuristics*, Mathematics of Computation **48**, no. 177 (1987), 123–137.

[CR] R. Cosset, D. Robert, *Computing (ℓ, ℓ)-isogenies in polynomial time on Jacobians of genus* 2 *curves*, `http://hal.inria.fr/docs/00/57/89/91/PDF/niveau.pdf`, 2011.

[DH] W. Diffie, M. Hellman, *New directions in cryptography*, Information Theory, IEEE Transactions on 22, no. 6 (1976), 644–654.

[DT] Diem, C., Thomé, E., *Index Calculus in Class Groups of Non-Hyperelliptic Curves of Genus Three*, Journal of Cryptology **21** (2008), no. 4, 591–611.

[Die] Diem, C., *An Index Calculus Algorithm for Plane Curves of Small Degree*, Algorithmic Number Theory, Lecture Notes in Comput. Sci. **4076**, Springer, Berlin (2006), 543–557.

[Die2] Diem, C., *Index Calculus in Class Groups of Non-Hyperelliptic Curves of Genus* 3 *from a Full Cost Perspective*, preprint (2006) `http://www.math.uni-leipzig.de/~diem/preprints/sharcs.pdf`.

[Dud] A. Dudeanu, Ph.D. Thesis, Ecole polytechnique fédérale de Lausanne.

[ElG] T. Elgamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, In Advances in Cryptology, pp. 10-18, Springer Berlin Heidelberg, 1985.

[Eng] Enge, A., *Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time*, Math. Comp. **71** (2002), no. 238, pp. 729–742.

[EG] Enge, A., Gaudry, P., *A general framework for subexponential discrete logarithm algorithms*, Acta Arith. **102** (2002), no. 1, pp. 83–103.

[FOR] Flon, S., Oyono, R., Ritzenthaler, C., *Fast addition on non-hyperelliptic genus* 3 *curves*, IACR Cryptology ePrint Archive 2004 (2004): 118.

[Gal] S. Galbraith, *Constructing isogenies between elliptic curves over finite fields*, LMS Journal of Computation and Mathematics **2** (1999), 118–138.

[Gal2] S. Galbraith, *Mathematics of public key cryptography*, Cambridge University Press, 2012.

[Gau] Gaudry, P., *An algorithm for solving the discrete log problem on hyperelliptic curves*, Advances in Cryptology - EUROCRYPT 2000, Lecture Notes in Comput. Sci. **1807**, Springer, Berlin (2000), 19–34.

[GS] P. Gaudry, E. Schost, *Construction of secure random curves of genus* 2 *over prime fields*, Advances in Cryptology-EUROCRYPT 2004, 239–256, Springer Berlin Heidelberg, 2004.

[GTTD] Gaudry, P., Thomé , E., Thériault, N., Diem, C., *A Double Large Prime Variation for Small Genus Hyperelliptic Index Calculus*, Math. Comp. **76** (2007), no. 257, 475–492.

[Har] R. Hartshorne, *Algebraic geometry*, Vol. 52, Springer Science & Business Media, 1977.

[Hes] Hess, F., *Computing Riemann-Roch spaces in algebraic function fields and related topics*, J. Symbolic Comput. **33** (2002), no. 4, 425–445.

[KL] J. Katz, Y. Lindell, *Introduction to modern cryptography*, CRC Press, 2014.

[Koh] D. Kohel, *Endomorphism rings of elliptic curves over finite fields*, Ph.D. thesis, University of California, 1996.

[Kob] N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of computation **48**, no. 177 (1987), 203–209.

[Kob2] Koblitz, N., *Hyperelliptic Cryptosystems*, Journal of Cryptology **1** (1989), 139–150.

[LL] K. Laine, K. Lauter, *Time-Memory Trade-offs for Index Calculus Attacks in Genus* 3, Preprint `http://eprint.iacr.org/2014/346`, 2014. Will appear in Journal of Mathematical Cryptology.

[LR] D. Lubicz, D. Robert, *Computing isogenies between abelian varieties*, Compositio Mathematica **148**, no. 05 (2012), 1483–1515.

[Mil] V. Miller, *Use of elliptic curves in cryptography*, In Advances in Cryptology—CRYPTO'85 Proceedings, Springer Berlin/Heidelberg, 1986, 417–426.

[Mil2] J. Milne, *Jacobian varieties*, Arithmetic geometry, Springer New York (1986), 167–212.

[Mum] D. Mumford, *Tata lectures on theta II: Jacobian theta functions and differential equations*, Progress in Mathematics **43** (1984).

[Mum2] D. Mumford, with appendices by C. P. Ramanujam, and J. I. Manin, *Abelian varieties*, Hindustan Book Agency, New Delhi, 2012.

[MW] W. Waterhouse, J. Milne, *Abelian varieties over finite fields*, Proc. Sympos. Pure Math. Vol. 20, 1971.

[Nag] Nagao, K., *Index Calculus Attack for Jacobian of Hyperelliptic Curves of Small Genus Using Two Large Primes*, Japan Journal of Industrial and Applied Mathematics, **24** (2007), 289–305.

[Pol] J. Pollard, *Monte Carlo methods for index computation* (*modp*), Mathematics of computation 32, no. 143 (1978), 918–924.

[Rob] D. Robert, *Fonctions thêta et applications à la cryptographie* (Ph.D. Thesis), Université Henri Poincaré, 2010.

[Rob2] D. Robert, *Computing cyclic isogenies using real multiplication* (Talk Notes), ANR Peace meeting, April 2013, Paris.

[Sem] I. Semaev, *Summation polynomials and the discrete logarithm problem on elliptic curves*, IACR Cryptology ePrint Archive 2004 (2004): 31, `http://eprint.iacr.org/2004/031`.

[Sem2] I. Semaev, *New algorithm for the discrete logarithm problem on elliptic curves*, IACR Cryptology ePrint Archive 2015 (2015): 310, `http://eprint.iacr.org/2015/310`.

[Shim] G. Shimura, *Abelian varieties with complex multiplication and modular functions*, vol. 46, Princeton University Press, 1998.

[Shio] T. Shioda, *On the graded ring of invariants of binary octavics*, American Journal of Mathematics (1967), 1022–1046.

[Sil1] J. Silverman, *The arithmetic of elliptic curves*, Vol. 106, Springer, 2009.

[Sil2] J. Silverman, *Advanced topics in the arithmetic of elliptic curves*, Vol. 151, Springer-Verlag, 1994.

[Smi] Smith, B., *Isogenies and the Discrete Logarithm Problem in Jacobians of Genus 3 Hyperelliptic Curves*, Advances in Cryptology - EUROCRYPT 2008, Lecture Notes in Comput. Sci. **4965**, Springer, Berlin (2008), 163–180.

[SWD] O. Schirokauer, D. Weber, T. Denny, *Discrete logarithms: the effectiveness of the index calculus method*, In Algorithmic number theory, pp. 337–361, Springer Berlin Heidelberg, 1996.

[The] Thériault, N., *Index calculus attack for hyperelliptic curves of small genus*, Advances in Cryptology - ASIACRYPT 2003, Lectures Notes in Comput. Sci. **2894**, Springer, Berlin (2003), 75–92.

[VJS] M. D. Velichka, M. J. Jacobson Jr., A. Stein, *Computing discrete logarithms in the Jacobian of high-genus hyperelliptic curves over even characteristic finite fields* Math. Comp. **83** (2014), 935–963.

[Was] L. C. Washington, *Elliptic curves: number theory and cryptography*, CRC press, 2008.

[Wen] A. Weng, *A class of hyperelliptic CM-curves of genus three*, Journal of the Ramanujan Math. Soc. **16**, no. 4 (2001), 339–372.

[Wen2] A. Weng, *Constructing hyperelliptic curves of genus 2 suitable for cryptography*, Mathematics of Computation **72**, no. 241 (2003), 435–458.

[Wes] B. Wesolowski, *Walking on Isogeny Graphs of Genus* 2 *Hyperelliptic Curves* (M.Sc. Thesis), Ecole polytechnique fédérale de Lausanne, 2014.

[ZLR] A. Zykin, G. Lachaud, C. Ritzenthaler, *Jacobians among abelian threefolds: A formula of Klein and a question of Serre*, Doklady Mathematics **81**, no. 2, SP MAIK Nauka/Interperiodica, 2010.