Trace Zero Varieties in Cryptography: Optimal Representation and Index Calculus

Maike Massierer

Trace Zero Varieties in Cryptography:
Optimal Representation and Index Calculus

Inauguraldissertation
zur
Erlangung der Würde eines Doktors der Philosophie

vorgelegt der
Philosophisch–Naturwissenschaftlichen Fakultät
der Universität Basel

von
Maike Massierer

aus
Nürnberg, Deutschland

Basel, 2014

Genehmigt von der Philosophisch–Naturwissenschaftlichen Fakultät
auf Antrag von

Prof. Dr. Elisa Gorla
Prof. Dr. Tanja Lange

Basel, den 10. Dezember 2013

Prof. Dr. Jörg Schibler,
Dekan

Trust in the Lord with all your heart, and do not lean on your own understanding.
In all your ways acknowledge him, and he will make straight your paths.

The Bible (ESV), Proverbs 3:5–6

Summary

The trace zero variety associated to an elliptic or hyperelliptic curve is an abelian variety defined over a finite field $\mathbb{F}_q$. Its $\mathbb{F}_q$-rational points yield a finite group, the trace zero subgroup of the degree zero Picard group of the original curve, consisting of all points of trace zero with respect to some field extension $\mathbb{F}_{q^n}|\mathbb{F}_q$ of prime degree n. This group has been proposed for use in cryptographic systems based on the discrete logarithm problem by Frey, since the group arithmetic is particularly fast, and for use in pairing-based cryptosystems by Rubin and Silverberg, since it produces particularly secure pairings. In this thesis, we study two aspects of using trace zero subgroups in cryptography: optimal-size representation of the elements and the hardness of the discrete logarithm problem.

For the efficient use of memory and bandwidth, one desires an optimal-size representation of the elements of trace zero subgroups, i.e. a representation whose size matches the size of the group. We propose two such representations. The first one builds on an equation for the trace zero subgroup of an elliptic curve that we derive from Semaev's summation polynomials. It can be made practical for small values of n. The second one is via the coefficients of a rational function, and it works for trace zero subgroups of elliptic and hyperelliptic curves of any genus, with respect to a base field extension of any prime degree. For each representation, we present efficient compression and decompression algorithms (to compute the representation, and to recover a full point from its representation), and complement them with implementation results. We discuss in detail the practically relevant cases of small genus and extension degree, and we compare with the other known compression methods of Naumann, Lange, and Silverberg. Both representations that we propose are compatible with scalar multiplication of points, and they are the first representations with this property.

We also investigate the hardness of the discrete logarithm problem in trace zero subgroups. For this purpose, we propose an index calculus algorithm to compute discrete logarithms in these groups, following the approach of Gaudry for index calculus in abelian varieties of small dimension. We make the algorithm explicit for small values of n and study its complexity as well as its practical performance with the help of our own Magma implementation. Finally, we compare this approach with other possible attacks on the discrete logarithm problem in trace zero subgroups and draw some general conclusions on the suitability of these groups for cryptographic systems.

Contents

List of algorithms

List of tables

List of notation

C	usually an elliptic or a hyperelliptic curve	10
$C(\mathbb{F})$	$\mathbb{F}$-rational points of the curve C	11
$[D]$	class of the divisor D in the Picard group	12
$D_1 \leq D_2$	comparison of two degree 0 divisors D_1, D_2	11
$D_1 \sim D_2$	equivalence of divisors, i.e. $[D_1] = [D_2]$	12
$D_1 \oplus D_2$	reduced divisor s.t. $D_1 \oplus D_2 \sim D_1 + D_2$, elliptic curve addition law	12
$\deg(D)$	degree of a divisor D	11
Div_C	divisor group of the curve C	11
$\mathrm{Div}_C(\mathbb{F}_{q^n})$	group of $\mathbb{F}_{q^n}$-rational divisors on C	11
Div_C^0	group of degree 0 divisors on C	11
$\mathrm{div}(h)$	principal divisor of a function h	11
d_{reg}	regularity index, or degree of regularity, of an ideal	24
E	usually an elliptic curve	12
e_i	i-th elementary symmetric polynomial	40
$\tilde{e}_i$	Weil restriction of $e_i(x, x^q, \ldots, x^{q^{n-1}})$	40
$\mathcal{F}$	factor base	27
f_m	m-th summation polynomial, or Semaev polynomial	36
$\tilde{f}_n$	Weil restriction of $f_n(x, x^q, \ldots, x^{q^{n-1}})$	39
$\mathbb{F}_q$	finite field with q elements	10
$\overline{\mathbb{F}}_q$	algebraic closure of the field $\mathbb{F}_q$	11
g	usually the genus of a curve	10
g_n	symmetrized n-th Semaev polynomial	40
h_D	in Chapters 5, 6, the rational function such that $\mathrm{div}(h_D) = \mathrm{Tr}(D)$	58
H_D	in Chapters 5, 6, the polynomial $H_D = h_D(h_D \circ w)$	59
Jac_C	Jacobian variety of the curve C	12
$[k]P$	scalar multiplication by k of the elliptic curve point P	14
$L_N(a, c)$	sub-exponential complexity	9
$\log_P Q$	base-P discrete logarithm of Q	8
$\ell_{P,Q}$	line passing through the points P, Q	80
$\mathrm{lt}(f)$	leading term of a polynomial f	22
$\mathrm{lt}(I)$	leading term ideal, or initial ideal, of an ideal I	23
m_{reg}	Castelnuovo–Mumford regularity of an ideal	24
n	usually the prime degree of a field extension $\mathbb{F}_{q^n} \vert \mathbb{F}_q$	18
$\tilde{O}$	"soft-O", O-notation that ignores logarithmic factors	9
$\mathcal{O}$	point at infinity of an elliptic or hyperelliptic curve	11

$\mathrm{ord}(P)$	order of P	8
φ	Frobenius map	11
Pic^0_C	Picard group, or degree zero divisor class group, of the curve C	12
$\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$	$\mathbb{F}_{q^n}$-rational divisor classes in Pic^0_C	12
$\mathrm{Pic}^0_C[k]$	k-torsion points of Pic^0_C	12
$\mathcal{R}$	representation of a group	20
$\mathrm{Res}_{L\|K} V$	Weil restriction of the variety V w.r.t. the extension $L\|K$	15
T_n	trace zero subgroup of $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$	19
Tr	trace map	18
$[u, v]$	Mumford representation of a divisor	13
$u^{\varphi}(x)$	application of φ to the coefficients of the polynomial $u(x)$	18
V_n	trace zero variety associated to a curve w.r.t. the extension $\mathbb{F}_{q^n}\|\mathbb{F}_q$	19
v_P	vertical line passing through the point P	80
w	hyperelliptic involution	11

Acknowledgements

I would like to thank…

…my advisor Elisa Gorla for her competent, friendly, and patient guidance in carrying out this PhD project. Her ideas, her insights, her enthusiasm, and her knowledge inspired me, motivated me, and gave me a fresh perspective again and again, and I could not have done this work without. Elisa always had an open door and an open ear for my questions and concerns, she spent an incredible amount of time discussing and working with me, she carefully read this manuscript, and we co-authored three articles. I learned a great deal from her about math, crypto, teaching, academia, and life in general. I sincerely appreciate her mentorship, her advice, and her interest in every aspect of my work and career.

…Tanja Lange for reading this thesis in such great detail, for numerous valuable comments and the resulting discussions, and for coming to Basel on the day of my defense.

…all those people who invited me to give talks and with whom I had helpful discussions about my work: Anja Becker, Andreas Enge, Florian Heß, Pierrick Gaudry, Damien Robert, Ben Smith, Peter Schwabe, Vanessa Vitse, and Bo-Yin Yang.

…Răzvan Bărbulescu for proofreading parts of this thesis.

…all my colleagues, office mates, and math friends, and in particular Alberto, Alex, Ambros, Andriy, Anna-Lena, Christian, Doris, Felix, Immu, Isac, Jung Kyu, Liza, Maria, Matey, Moritz, Peter, Răzvan, Roland, and Susanna, for discussions about math and many other things, for help with abstract and real-life problems, and for the fun times we spent together inside and outside of various math and computer science departments.

…Richard Buckland, Christine Flaig, Jim Franklin, Florian Heß, and Joachim Rosenthal, who have taught me so much about mathematics, computer science, cryptography, and life, and who have supported and encouraged me throughout the years.

…the math department of the University of Zürich for access to their computing facilities, and Carsten Rose and Rafael Ostertag for their competent and friendly support.

…my brother Björn for late night help with layout, design, and technical questions on several occasions.

…my family for their love, encouragement, support, and prayers, and especially my Mom for always being there for me.

I gratefully acknowledge financial support of the Swiss National Science Foundation throughout my PhD studies. I am also thankful for several travel stipends that allowed me to attend schools and workshops all over the world, including the ECC Workshops, a number of ECRYPT II events, many meetings organized by the Swiss Doctoral Program in Mathematics, SP-ASCrypto in São Paulo, Trends in Coding Theory in Ascona, and an Oberwolfach Seminar.

Chapter 1

Introduction

It is hard to imagine how today's global society could function without electronic communication via the Internet, mobile phones, and the like. Clearly, the recent rapid development of technologies for digital message transmission has not only brought a welcome increase in speed, reliability, and availability, but also a great need for protection of the privacy of digital communication. The goal of *public key cryptography* is to achieve exactly this. It provides methods for secure digital communication such as data encryption, digital signatures, and authentication techniques. They are implemented in numerous applications used in modern everyday life, such as online shopping and banking websites, ATMs, mobile phone connections, and many European passports, in order to protect the data and identity of the users.

Cryptographic systems are usually designed with respect to computationally hard problems in the sense that "breaking" the system (e.g. decrypting data without possessing the private key, forging a signature, or authenticating as someone else) is shown to be computationally at least as hard as solving a given problem. Then, the system is secure if the problem is computationally hard, meaning that no "efficient" algorithm to solve the problem exists.

Although a number of supposedly hard problems have been proposed, only two are widely deployed in practice. The first is factoring large integers, and the corresponding cryptosystem is called RSA. The second is the *discrete logarithm problem*, or *DLP*: Given an element h of a finite cyclic group $G = \langle g \rangle$, the task is to compute a number $\ell \in \mathbb{Z}/|G|\mathbb{Z}$ such that $g^\ell = h$. This number is the base-g *discrete logarithm* of h. The most common cryptosystems that base their security on the DLP are Diffie–Hellman key exchange, ElGamal, and the Digital Signature Algorithm DSA. In fact, it was the idea of the key exchange algorithm of Diffie and Hellman that started the field of public key cryptography and first introduced the DLP in multiplicative groups of finite fields in 1976 [DH76].

A combination of the Pohlig–Hellman algorithm and the Pollard–Rho algorithm can solve the DLP in any given group G in time $O(\sqrt{N})$, where N is the largest prime factor of $|G|$. This is called a *generic attack* (since it does not use any specific properties of the group) or *square root attack*, and its complexity is exponential in the size of the input (i.e. $\log |G|$). With regard to this attack, groups of prime order have the strongest DLP, and for this reason the order of a group used in a DLP-based cryptosystem must be efficiently computable, so that one can decide whether it has a sufficiently large prime factor.

By exploiting the specific properties of a group and its operation, discrete logarithm algorithms can often achieve much better complexities than the Pollard–Rho algorithm. For example, computing discrete logarithms is trivial in the additive group $\mathbb{Z}/p\mathbb{Z}$ for a prime p because one can invert h. Hence the actual hardness of the DLP depends crucially on the underlying group. The best candidates for cryptosystems are those groups where the DLP is the hardest, in an ideal case no algorithm of complexity better than $O(\sqrt{|G|})$ is known.

In order to obtain efficient cryptosystems, it is also important that the elements of the group G are conveniently representable and that the group operation is efficiently implementable on a computer.

Summarizing, a major goal in public key cryptography is to identify and study groups that satisfy the following conditions (see also [Lan01]).

- There is a convenient representation of the group elements.
- Computation with the group elements is efficient.
- The group order is efficiently computable.
- There is no attack on the DLP in the group which has better complexity than generic algorithms.

The most popular groups in this context are multiplicative groups of finite fields, but they do not provide optimal security, since there exist sub-exponential attacks on the DLP, namely the number field sieve and the function field sieve. Such attacks are asymptotically better than generic (square root) attacks, which have exponential complexity. For some fields of small characteristic, an even better attack of quasi-polynomial complexity has recently been found, see [BGJT13].

A popular alternative are groups of $\mathbb{F}_q$-rational points of *elliptic curves* defined over finite fields $\mathbb{F}_q$, which were first proposed for use in DLP-based cryptosystems by Koblitz [Kob87] and Miller [Mil85] independently in 1985. For most such groups (in particular when q is prime), no attacks are known that have better complexity than generic algorithms. A natural generalization, suggested by Koblitz [Kob89] in 1989, is to use *Picard groups of hyperelliptic curves* defined over finite fields, but only curves of small genus are interesting (most popular is genus 2), since index calculus attacks can solve the DLP in sub-exponential complexity for curves of medium and large genus.

A powerful tool in this context are *pairings*, i.e. bilinear maps defined on a product of elliptic curves (or more general abelian varieties) and mapping into a finite field. They were first introduced to cryptography by Menezes, Okamoto, and Vanstone [MOV93] and Frey and Rück [FR94], who used the Weil and Tate pairings, respectively, to reduce the DLP in Picard groups of supersingular elliptic and hyperelliptic curves to the DLP in finite fields. A number of positive applications of pairings in cryptography were subsequently found, the most prominent ones being tripartite Diffie–Hellman key exchange by Joux [Jou00] and identity-based encryption invented by Boneh and Franklin [BF03] and independently by Sakagi, Ohgishi, and Kasahara [SOK01].

The trace zero variety. In this thesis, we study the *trace zero subgroup* of the Picard group of an elliptic or hyperelliptic curve, which is yet another promising candidate in the context of DLP-based and pairing-based cryptography. Given a curve defined over a finite field $\mathbb{F}_q$ and a field extension $\mathbb{F}_{q^n}|\mathbb{F}_q$ of prime degree n, the trace zero subgroup consists of all $\mathbb{F}_{q^n}$-rational divisor classes of trace zero, i.e. of all $[D]$ such that

$$D + \varphi(D) + \ldots + \varphi^{n-1}(D) \sim 0,$$

where φ is the Frobenius endomorphism. The trace zero subgroup can be realized as the $\mathbb{F}_q$-rational points of the *trace zero variety*, an $(n-1)g$-dimensional abelian variety defined over $\mathbb{F}_q$ built by Weil restriction from the original curve of genus g.

The trace zero variety was first proposed in the context of cryptography by Frey [Fre99] and further studied by Naumann [Nau99], Weimerskirch [Wei01], Blady [Bla02], Lange [Lan01, Lan04b], Diem and Scholten [Die01, DS03, DS], Rubin and Silverberg [Sil05, RS09], and Avanzi and Cesena [AC07, Ces08, Ces10]. Although the trace zero subgroup is a proper subgroup of the $\mathbb{F}_{q^n}$-rational points of the Jacobian of the curve, it can be shown that the DLP in both groups has the same complexity. Therefore, from a mathematical point of view, trace zero variety cryptosystems may be regarded as the (hyper)elliptic curve analog of torus-based cryptosystems such as LUC [SS95], Gong–Harn [GH99], XTR [LV00], and CEILIDIH [RS03].

The trace zero subgroup is of particular interest in the context of pairing-based cryptography. Rubin and Silverberg have shown in [RS02, RS09] that the security of pairing-based cryptosystems can be improved by using supersingular abelian varieties of dimension greater than one in place of elliptic curves. Jacobians of hyperelliptic curves and trace zero varieties are therefore the canonical examples for such applications. E.g., over a field of characteristic 3 the known examples of groups with highest security parameter (i.e. 7.5) come from trace zero subgroups relative to a field extension $\mathbb{F}_{q^5}|\mathbb{F}_q$.

Scalar multiplication in the trace zero subgroup is particularly efficient, due to a speed-up using the Frobenius endomorphism, see [Lan04b, AC07]. This technique is similar to the one used on Koblitz curves [Kob91] and has afterwards been applied to GLV/GLS curves [GLV01, GLS11], which are the basis for several recent implementation speed records for elliptic curve arithmetic [LS12, FHLS13, BCHL13]. In [AC07], Avanzi and Cesena show that trace zero subgroups in general deliver better scalar multiplication performance than elliptic curves and trace zero subgroups constructed from elliptic curves over degree 5 extension fields are almost three times faster than elliptic curves for the same group size. They conclude that trace zero subgroups are very interesting groups for the design of cryptographic systems based on the discrete logarithm problem due to their vastly superior performance, but that such systems sacrifice some memory and bandwidth.

Efficient representation. The first topic of this thesis is to solve this problem by developing two representations for the elements of trace zero subgroups which are both efficiently computable and optimal in size.

There is also another motivation for studying compact representations of the trace zero elements: The hardness of the discrete logarithm problem in a group is closely connected with the size of the representation of the group elements. Usually, the hardness of the DLP is measured as a function of the group size. However, for practical purposes, the comparison with the size of the representation of group elements is a better indicator, since it quantifies the storage and transmission costs connected with using the corresponding cryptosystem. Therefore, in order to make the comparison between DLP complexity and group size a fair one, we are interested in a compact representation that reflects the size of the group.

An optimal-size representation for elliptic curves is well known. In the cryptographic setting, it is standard procedure to represent an elliptic curve point by its x-coordinate only, since the y-coordinate can easily be recomputed, up to sign, from the curve equation. If desired, the sign can be stored in one extra bit of information. Representing a point via its x-coordinate gives an optimal representation for the elements of the group of $\mathbb{F}_{q^n}$-rational points of an elliptic curve: Each of the approximately q^n points can be represented by one element of $\mathbb{F}_{q^n}$, or n elements of $\mathbb{F}_q$ after choosing a basis of the field extension, i.e. the representation has size $\log_2 q^n$ bits. Notice moreover that storing the sign of the y-coordinate is unnecessary, since this representation is compatible with scalar multiplication of points (the only operation needed in many DLP-based cryptosystems, e.g. Diffie–Hellman key exchange): For any $k \in \mathbb{Z}$, the x-coordinates of the points $[k]P$

and $-[k]P$ coincide. Not storing the sign of y amounts to working with equivalence classes $\{\pm P\}$ of points.

The standard compact representation for hyperelliptic curves is a generalization of this: A divisor class is represented by the u-polynomial of the Mumford representation, see [HSS01, Sta04]. Again, either one stores the signs of the coefficients of the v-polynomial in some extra bits, or one works with equivalence classes.

Since the trace zero subgroup has about $q^{(n-1)g}$ elements, an optimal-size representation should consist of approximately $\log_2 q^{(n-1)g}$ bits. A natural approach would be to represent an element of the trace zero subgroup via $(n-1)g$ elements of $\mathbb{F}_q$. For practical purposes, it is important that the representation can be efficiently computed ("compression") and that the original point can be easily recovered, possibly up to some small ambiguity, from the representation ("decompression").

Such representations and algorithms have been proposed by Naumann [Nau99, Chapter 4.2] for trace zero subgroups of elliptic curves and by Lange [Lan04b] for trace zero varieties associated to hyperelliptic curves of genus 2, both with respect to cubic field extensions, and by Silverberg [Sil05] for trace zero subgroups of elliptic curves with respect to base field extensions of degree 3 and 5. A compact representation for Koblitz curves has been proposed by Eagle, Galbraith, and Ong [EGO11]. They all build on the standard compact representation for elliptic and hyperelliptic curves mentioned above.

A common approach to compact representations is analogous to the one for elliptic curves: One finds a suitable equation (or several equations) for the variety, one drops one or several coordinates, and one uses the equation(s) to recompute it or them. Naumann, Lange, and Silverberg all follow this approach.

For this reason, after giving a short overview of the mathematical background needed in this thesis in *Chapter 2*, we study equations for the trace zero variety of elliptic curves in *Chapter 3*. Here and throughout the thesis, we put particular emphasis on the cases $n = 3, 5$, which are most relevant in practice. First we explain how to write rather straightforward equations in the Weil restriction coordinates $x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}$, which was already done for $n = 3$ by Frey [Fre99], Naumann [Nau99], and Diem [Die01]. Next, using the Semaev polynomial [Sem04], we derive an equation only in the x-coordinates that is compatible with the standard representation for elliptic curves, i.e. with dropping the y-coordinate. We show that this equation describes the points of the trace zero subgroup (though possibly not the points of the entire trace zero variety), up to some well-understood exceptions. This equation leads to the representation of Naumann for $n = 3$. It is useful not only for an efficient representation, but also for the index calculus attack we describe in Chapter 7 (see below), but its drawback is that its degree grows quickly with n. By symmetrizing the Semaev polynomial, we derive an equation of lower degree. It is also compatible with dropping the y-coordinate, and it identifies Frobenius conjugates of trace zero points.

We use this last equation to propose a new optimal representation for the points of the trace zero variety of an elliptic curve in *Chapter 4*. Similarly to the approach of [EGO11], our method identifies Frobenius conjugates, i.e. it yields equivalence classes

$$\{\pm\varphi^i(P) \mid i = 0, \ldots, n-1\}.$$

We also give compression and decompression algorithms and a detailed comparison of our method with those of Naumann for $n = 3$ and Silverberg for $n = 3, 5$. We conclude that our algorithms are more efficient than those of [Nau99, Sil05] and points are recovered with smaller ambiguity. In addition, our representation is (to the extent of our knowledge) the only one that is compatible with scalar multiplication of points, since for any $k \in \mathbb{Z}$ and $i \in \{0, \ldots, n-1\}$, we have $[k]\varphi^i(P) = \varphi^i([k]P)$.

In *Chapter 5* we present a different new optimal-size representation for the elements of the trace zero subgroup associated to an elliptic or hyperelliptic curve of any genus g and any field extension of prime degree n. It is the first representation that works for elliptic curves with $n > 5$, for hyperelliptic curves of genus 2 with $n > 3$, and for hyperelliptic curves of genus $g > 2$. The basic idea is to represent a given divisor class via the coefficients of the rational function whose associated principal divisor is the trace of the given divisor. This representation also identifies well-defined equivalence classes of points, and scalar multiplication is well-defined on such classes. We give compression and decompression algorithms, and we show that these algorithms are comparably or more efficient than all previously known compression and decompression methods and that they are efficient even for medium to large values of g and n.

In *Chapter 6*, we specialize the results of Chapter 5 to trace zero subgroups of elliptic curves, focussing in particular on possible simplifications compared to the more general case and also giving some self-contained and simpler proofs. We see that the algorithms are particularly efficient here and give a detailed comparison with the other compression methods for trace zero subgroups of elliptic curves.

Index calculus. In view of all the applications of trace zero subgroups discussed above, a detailed assessment of the complexity of the discrete logarithm problem becomes necessary. The second goal of this thesis is to do this by means of developing an index calculus algorithm, following the approach of Gaudry [Gau09] for index calculus in abelian varieties. Since the DLP in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$, where Pic^0_C is the Picard group of an elliptic or hyperelliptic curve defined over $\mathbb{F}_q$, has the same complexity as the DLP in the corresponding trace zero subgroup, the results are also relevant to groups $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$.

Index calculus is a classical method to compute discrete logarithms in groups where a concept of prime elements and factorization is available. After choosing a set of small primes, the so-called *factor base*, the algorithm performs two computationally costly steps. First, it chooses a random group element and checks whether all its prime factors are elements of the factor base. This is not very likely, but if it is true, the algorithm has found a *relation*. By trying sufficiently many random group elements, the algorithm generates about as many relations as there are factor base elements. In the second step, the algorithm solves a sparse linear system, obtained from the relations. By doing this, it deduces the discrete logarithms of all factor base elements, and they may then be used to compute the desired discrete logarithm.

It is an important innovation of Gaudry's index calculus algorithm that it works in abelian varieties, where no concept of factorization is available a priori. His idea is to translate the condition for a relation into a system of polynomial equations and to solve this system using Gröbner basis techniques. For this approach, it is important that a convenient representation of the elements and simple equations of the variety are available. Gaudry's algorithm has thus far been applied to algebraic tori by Granger and Vercauteren [GV05] and the Weil restriction of elliptic and hyperelliptic curves defined over extension fields by Gaudry himself, by Joux and Vitse, and by Nagao [Gau09, JV12, Nag10]. An analogous algorithm for elliptic curves has been proposed independently by Diem [Die11].

In *Chapter 7*, we apply Gaudry's index calculus algorithm to the trace zero variety associated to an elliptic curve. We give a detailed description of the algorithm, using the equations for the trace zero subgroup derived in Chapter 3 and making explicit the choice of the factor base and the shape of the polynomial systems. We perform a heuristic complexity analysis of the algorithm similar to the one of Gaudry, but asymptotic in both q and n. When regarding n as a constant, as Gaudry does, the attack has complexity $\tilde{O}(q^{2-2/(n-1)})$, which means that it is better than generic (square root) attacks on the trace zero subgroup for $n > 3$. However, the complexity depends exponentially on n, which is due to the fact that n determines the size of the polynomial system, and Gröbner basis computations are exponential in this number. This shows that the index calculus attack is only feasible for very moderate values of n.

We furthermore study the practicality of the attack using Magma. Our experiments indicate that it is possible to compute a discrete logarithm in a 60-bit trace zero subgroup when $n = 3$ and that it is already extremely difficult to solve one single polynomial system when $n = 5$. By employing a trick from [JV12] and using a hybrid approach to solve the system (see [YCC04, BFP08]), we are able to generate enough relations to compute a discrete logarithm for a 36-bit trace zero subgroup for $n = 5$, but this method does not seriously threaten the DLP in this group.

Finally, we compare the index calculus attack to other known attacks, namely the Pollard–Rho algorithm, index calculus algorithms on the whole curve, and cover attacks, and we discuss the suitability of trace zero subgroups in general for DLP-based and pairing-based cryptosystems. Our conclusion is that there exist no attacks on the DLP in trace zero subgroups of elliptic curves with $n = 3$ that are better than generic (square root) attacks. The security of trace zero subgroups of elliptic curves with $n \geq 5$ and of hyperelliptic curves is not optimal, since Gaudry's index calculus algorithm is better than generic attacks here, but we have seen that the Gröbner basis computations that are part of this approach become infeasible very quickly. It is our impression that further study, in particular on specialized Gröbner basis algorithms, is needed in order to make this attack a real-life threat. We also see that the attack does not threaten the security of pairing-based cryptosystems using the trace zero variety, provided that the curve is chosen appropriately.

Experiments and equations. We have implemented all algorithms presented in this thesis in Magma [BCP97]. These implementations are neither specialized nor optimized and serve mainly as a proof of concept. For the most frequently studied cases $g = 1, n = 3, 5$ and $g = 2, n = 3$, we often give timings, which serve as an indication for the feasibility and efficiency of our methods. All computations were performed on one core of an Intel Xeon X7550 Processor (2.00 GHz) on a Fujitsu Primergy RX900S1.

In Chapter 3, we explain how to compute explicit equations, but many of them are too long to be printed in this thesis, even in very small font. These equations can be found online at `http://maikemassierer.wordpress.com/phdthesis`.

Chapter 2

Preliminaries

We give the necessary mathematical background for this thesis. We try to be self-contained but brief, and we state the relevant results without proof. In many cases, we take a very concrete point of view and treat objects and techniques from algebra, number theory, and algebraic geometry from a computational perspective, as needed in cryptography. In each section, we point out references that give a more general mathematical treatment of the subject. The *Handbook of Elliptic and Hyperelliptic Curve Cryptography* [ACD$^+$06], which covers all topics of this chapter—though often also without proofs, contains a more comprehensive list of references to the relevant literature. For a general introduction to cryptography, we refer the reader to the book of Stinson [Sti06].

2.1 Public key cryptography and the discrete logarithm problem

The main goal of classical cryptography was the encryption of messages, in order to make their content unreadable for unauthorized third parties. This was achieved with a *symmetric* setup, where the communicating parties share a common *secret key*, and this key is used for both encryption and decryption.

Modern cryptography has a much wider range of applications, including not only data encryption but also digital signatures, authentication methods, key distribution and key agreement protocols, interactive proof systems, secret sharing schemes, and many others (see [Sti06]). Generally speaking, all of these techniques are needed to allow secure and private digital communication by protecting the data and the identities of the communicating parties in the presence of (possibly malicious) third parties.

While symmetric techniques are still important today, *asymmetric* techniques provide particularly powerful tools, since they do not require the prior agreement on a shared secret key. Instead, asymmetric cryptography, also called *public key cryptography*, operates with a pair of keys to perform opposite tasks: The *public key* is used for encryption of a message or verification of a digital signature, and the *private key* is used for decryption of a message or creation of a digital signature. Public key cryptography also provides techniques for agreeing on a shared secret (e.g. a secret key to be used subsequently for symmetric encryption) via an insecure channel (e.g. the Internet). In fact, the publication of the first such algorithm in 1976 [DH76] marked the birth of public key cryptography in the public scientific community. This algorithm is the *Diffie–Hellman key exchange protocol*, which (together with its variants) is still the most widely deployed key agreement algorithm today. The most common public key encryption algorithms are the RSA [RSA78] and ElGamal

[ElG85] cryptosystems (and their variants), and the standard for digital signatures is the Digital Signature Algorithm (DSA) [Nat94], which is a variant of the ElGamal signature scheme [ElG85].

Diffie–Hellman key exchange and both ElGamal systems base their security on the *discrete logarithm problem* in the sense that solving the discrete logarithm problem is sufficient for "breaking" each of the systems. In the case of Diffie–Hellman, this means that the shared secret can be computed just from the information sent over the channel. In the case of ElGamal, it means that an encrypted message can be decrypted or a signature can be forged without possessing the private key.

Definition 2.1. Let G be a finite group (here written additively). Given two elements $P \in G$ and $Q \in \langle P \rangle$, the *discrete logarithm problem (DLP)* is

$$\text{find an element } \ell \in \mathbb{Z}/(\operatorname{ord} P)\mathbb{Z} \text{ such that } \ell P = Q.$$

The number ℓ is called the *base-P discrete logarithm of* Q and denoted by $\log_P Q$.

Diffie–Hellman key exchange and the ElGamal systems can only be *secure* if the DLP is computationally *hard*, meaning that no *efficient* algorithm exists to solve it (this notion is made more precise at the end of this section). Such cryptosystems are called *DLP-based* cryptosystems, since they base their security on the hardness of the DLP. Since it is extremely difficult to show that no efficient algorithm to solve the DLP *exists*, a DLP-based cryptosystem is usually considered secure as long as no efficient algorithm to solve the DLP is *known*. An algorithm that solves the DLP, or more generally a hard problem on which the security of a cryptosystem relies, is called an *attack*.

The actual hardness of the DLP depends crucially on the group G. For example, the DLP is very easy in the additive group $\mathbb{Z}/p\mathbb{Z}$, since it can be solved there by a simple division. Therefore, it makes sense only to speak of the hardness of the DLP *in a certain group*, and only groups where the DLP is hard are suitable for use in cryptography.

Remark 2.2. Breaking Diffie–Hellman or ElGamal in the sense explained above is actually equivalent (see [Sti06]) to solving the *Diffie–Hellman problem (DHP)*, which is

$$\text{given } P, aP, bP \in G, \text{ compute } abP.$$

Obviously, the Diffie–Hellman problem can be solved by solving a discrete logarithm problem (i.e. by computing either a from aP or b from bP). Whether the DLP can vice versa be reduced to the DHP, however, is an open question, and the two problems are known to be equivalent only in certain groups [Boe90, Mau94]. Nevertheless, cryptography extensively studies the hardness of the DLP, and not the DHP, and we also concentrate only on the DLP.

As explained in the Introduction, in order to build secure and practical DLP-based cryptographic systems, one actually needs groups that fulfill a number of requirements, the most important ones being that there is a convenient representation of the group elements, that the group operation can be implemented efficiently, that the group order is efficiently computable, and that the DLP in the group is hard. Elliptic curves and Picard groups of hyperelliptic curves of genus 2 defined over finite fields (see Section 2.2) are considered by many to be the best such groups, since arithmetic is efficient and they are currently conjectured to provide the strongest instances of the DLP. Naturally, subgroups of these groups are also good candidates (this idea goes back to Schnorr [Sch89], who was the first to propose using prime order subgroups of $\mathbb{F}_p^\times$). One such subgroup is the *trace* *zero subgroup* (see Section 2.4), and the goal of this thesis is to study the suitability of this group for DLP-based cryptosystems.

Before ending this section, we give some rather informal and intuitive notions concerning the efficiency of algorithms. For more detailed and precise definitions on this topic, see [CLRS09].

The efficiency of an algorithm is measured by its *complexity* in terms of the size of the input. For algorithms that solve the DLP, this is $\log_2 N$ if $N := |\langle P \rangle| = \operatorname{ord}(P)$, i.e. the size of the group we are computing in.

An algorithm is called *polynomial-time* if it has polynomial complexity in $\log N$, i.e. of the form $O(f(\log N))$ where f is a polynomial. Such algorithms are considered efficient. Therefore, cryptosystems based on problems that can be solved by polynomial-time algorithms are not secure. An example for this is the DLP in $\mathbb{Z}/p\mathbb{Z}$.

An algorithm is called *exponential-time* if it has exponential complexity in $\log N$, i.e. of the form $O(2^{\log N}) = O(N)$. Such algorithms are very inefficient, and cryptosystems based on hard problems for which only exponential-time algorithms are known are generally considered secure. For example, since the DLP is a finite problem, it can always be solved by *exhaustive search*. However, this involves trying out all elements of $\langle P \rangle$, which has complexity $O(N)$ and is therefore exponential-time. Hence the existence of such a brute force algorithm does not threaten the security of a cryptosystem. The reasoning behind this is that the group can always be chosen large enough (say of size 2^{100}) to make such attacks infeasible. Examples of groups where only exponential DLP-algorithms are currently known are elliptic curves defined over finite prime fields and Picard groups of hyperelliptic curves of genus 2 defined over finite fields.

Sub-exponential algorithms have complexity

$$L_N(a,c) = \exp\left((c + o(1))(\log N)^a (\log\log N)^{1-a}\right), \quad 0 < a < 1,$$

which truly lies in between exponential (this would be $a = 1$) and polynomial (this would be $a = 0$). Cryptosystems that base their security on a problem for which a sub-exponential attack is known are considered less secure, but they are still used in practice. This must be compensated for by choosing a larger group. Examples of such groups in the context of the DLP are multiplicative groups of finite prime fields and Picard groups of hyperelliptic curves of large genus defined over finite fields. This explains why many experts consider cryptosystems that operate in groups of rational points on elliptic curves superior to cryptosystems that operate in multiplicative groups of finite fields: Since the finite field DLP is weaker than the elliptic curve DLP, finite field cryptosystems must use larger fields than elliptic curve cryptosystems. Therefore, finite field cryptosystems use more memory and bandwidth, and in particular, they have larger keys than elliptic curve cryptosystems.

A common notation in connection with the O-notation is $\tilde{O}$, often called "soft-O". It disregards logarithmic factors, i.e. we write $f(x) = \tilde{O}(g(x))$ when

$$f(x) = O(g(x) \log^k g(x))$$

for some number k, some function g, and some input size x.

2.2 Elliptic and hyperelliptic curves

Elliptic curves were proposed for the use in DLP-based cryptosystems by Koblitz [Kob87] and Miller [Mil85] independently in 1985. Since elliptic curves defined over finite fields yield groups that have both fast arithmetic and a secure DLP, they are widely used in cryptography today. Hyperelliptic curves are a generalization of elliptic curves, and Koblitz [Kob89] first suggested them for DLP-based cryptosystems in 1989.

Although classically elliptic curves are not considered to be hyperelliptic curves, they can be seen as hyperelliptic curves of genus 1. In this section, we present a unified treatment, as is common in cryptography. We put particular emphasis on pointing out how elliptic curves relate to hyperelliptic curves, where the differences lie, and where elliptic curves are simpler (see especially Remark 2.7).

One major difference is that the points on an elliptic curve form a group, and finite subgroups thereof may be used directly in cryptosystems, which is not true for hyperelliptic curves. Instead, DLP-based hyperelliptic curve cryptosystems work in the *Picard group* (see Definition 2.5 (viii)), which is a well-studied group associated to any hyperelliptic curve. Arithmetic in this group is efficient, and the DLP is conjectured to be hard when the genus of the curve is sufficiently small. The degree zero Picard group of an elliptic curve is isomorphic to the curve itself.

We briefly recall the most important definitions and facts and introduce important notation for the rest of this thesis. Washington [Was08] and Menezes, Wu, and Zuccherato [MWZ98] cover elliptic and hyperelliptic curves with a view towards cryptographic applications, a more general exposition can be found in the book of Silverman [Sil09]. Good references on algebraic curves are the books of Fulton [Ful08] and Stichtenoth [Sti93], the latter is written in the language of function fields.

Let $\mathbb{F}_q$ be a finite field of cardinality q, where q is some prime power. *Whenever we deal with elliptic and hyperelliptic curves, we assume that $\mathbb{F}_q$ does not have characteristic 2.* This is only for convenience, since in this case, the curves can be described by equations of a particularly simple shape, as given below in Definition 2.3. All results of this thesis, however, can be generalized to the case of arbitrary positive characteristic. We comment on this with respect to each topic in the corresponding chapters.

We consider only *imaginary* hyperelliptic curves, since they are the ones relevant for cryptography. Thus in this whole thesis, *we use the term hyperelliptic curve to mean an imaginary hyperelliptic curve.*

Definition 2.3. Let $\mathbb{F}_q$ be a finite field of odd characteristic, let $g \geq 1$, and let C be a projective curve defined by an affine equation

$$C : y^2 = f(x), \tag{2.1}$$

where $f \in \mathbb{F}_q[x]$ is a monic polynomial of degree $2g+1$ that has no multiple zeros. If $g = 1$, then C is an *elliptic curve*, and if $g \geq 2$, then C is a *hyperelliptic curve of genus g*, both defined over $\mathbb{F}_q$.

It is obvious from this definition that elliptic curves can be seen as hyperelliptic curves of genus 1. The condition that f has no multiple zeros implies that an elliptic curve is smooth and that the affine part of a hyperelliptic curve is smooth.

Remark 2.4. Let E be an elliptic curve. When $\mathbb{F}_q$ does not have characteristic 3, a simple transformation yields an equation of *short Weierstraß form*

$$E : y^2 = x^3 + Ax + B,$$

which we will often use to write explicit formulas. In this case, smoothness is equivalent to $4A^3 + 27B^2 \neq 0$.

Elliptic and hyperelliptic curves are algebraic varieties, and we use the usual terminology: Let $\overline{\mathbb{F}}_q$ denote the algebraic closure of $\mathbb{F}_q$. The points on C are all points $(X,Y) \in \overline{\mathbb{F}}_q \times \overline{\mathbb{F}}_q$ that satisfy equation (2.1)—these are called the *affine points*—together with the *point at infinity*, which is denoted by $\mathcal{O}$. It is given by the projective coordinates $(0:1:0)$, which satisfy the homogenization of equation (2.1). For any algebraic extension field $\mathbb{F} \subseteq \overline{\mathbb{F}}_q$ of $\mathbb{F}_q$,

$$C(\mathbb{F}) = \{(X,Y) \in \mathbb{F} \times \mathbb{F} \mid Y^2 = f(X)\} \cup \{\mathcal{O}\}$$

is the set of $\mathbb{F}$-*rational* points of C. In particular, the point at infinity is $\mathbb{F}$-rational for every $\mathbb{F}$, and $C = C(\overline{\mathbb{F}}_q)$.

Definitions and Notation 2.5. Let C be an elliptic or hyperelliptic curve defined over $\mathbb{F}_q$.

(i) The *(hyper)elliptic involution* on C is

$$w : C \to C, \quad (X,Y) \mapsto (X,-Y), \quad \mathcal{O} \mapsto \mathcal{O}.$$

(ii) The *Frobenius map* on C is defined as

$$\varphi : C \to C, \quad (X,Y) \mapsto (X^q, Y^q), \quad \mathcal{O} \mapsto \mathcal{O}.$$

For any $n \geq 1$, the points of $C(\mathbb{F}_{q^n})$ are exactly the points of C which are fixed by φ^n.

(iii) A *divisor* D on C is a formal sum of points

$$D = \sum_{P \in C} m_P P$$

where only finitely many $m_P \in \mathbb{Z}$ are non-zero. In a natural way, these divisors form a group, the *divisor group* Div_C of C. Both w and φ extend to group homomorphisms on Div_C.

(iv) Let $\mathbb{F}_{q^n}, n \geq 1$ be a finite extension of $\mathbb{F}_q$. A divisor D is called $\mathbb{F}_{q^n}$-*rational* if

$$\varphi^n(D) = D.$$

The $\mathbb{F}_{q^n}$-rational divisors form a subgroup of Div_C denoted by $\mathrm{Div}_C(\mathbb{F}_{q^n})$. We write $\mathrm{Div}_C(\overline{\mathbb{F}}_q) = \mathrm{Div}_C$.

(v) The *degree* of D is defined as

$$\deg D = \sum_{P \in C} m_P.$$

The group of *divisors of degree zero* is denoted by Div_C^0 and is a subgroup of Div_C.

(vi) Let $D_1 = a_1P_1 + \ldots + a_kP_k - a\mathcal{O}, D_2 = b_1P_1 + \ldots + b_kP_k - b\mathcal{O} \in \mathrm{Div}_C$, $a_i, b_i, a, b \in \mathbb{Z}_{\geq 0}$, be two divisors of degree zero. If $a_i \leq b_i$ for all i we write $D_1 \leq D_2$.

(vii) Let $h \in \overline{\mathbb{F}}_q(C)$ be a function on C. Then the *principal divisor* of h is defined as

$$\mathrm{div}(h) = \sum_{P \in C} \mathrm{ord}_P(h) P,$$

where $\mathrm{ord}_P(h) \in \mathbb{Z}$ is the *order* of h at P. Principal divisors have degree 0. The *group of principal divisors* Prin_C is a subgroup of Div_C^0.

(viii) The *degree zero Picard group* of C is

$$\mathrm{Pic}_C^0 = \mathrm{Div}_C^0 \,/\, \mathrm{Prin}_C\,.$$

This is also called the *degree zero divisor class group* of C. For brevity, we write *Picard group*. For any $D, D_1, D_2 \in \mathrm{Div}_C^0$, we write $[D]$ for the equivalence class of D in Pic_C^0 and $D_1 \sim D_2$ for $[D_1] = [D_2]$. Furthermore, we write $\mathrm{Pic}_C^0(\mathbb{F}_{q^n})$ for the subgroup of $\mathbb{F}_{q^n}$-rational divisor classes, and $\mathrm{Pic}_C^0(\overline{\mathbb{F}}_q) = \mathrm{Pic}_C^0$.

(ix) For $k \geq 1$, we denote by

$$\mathrm{Pic}_C^0[k] = \{[D] \in \mathrm{Pic}_C^0 \mid k[D] = 0\}$$

the *k-torsion points* of Pic_C^0.

Hyperelliptic curve cryptosystems work in the Picard group $\mathrm{Pic}_C^0(\mathbb{F}_{q^n})$ of C for some $n \geq 1$. The group has approximately q^{ng} elements for large q, by the Theorem of Hasse–Weil (see [Sti93, Theorem V.2.3]).

Cryptographic literature often mentions the *Jacobian variety*. The Jacobian of C is the abelian variety Jac_C that contains C and has the property that for every algebraic extension field $\mathbb{F}$ of $\mathbb{F}_q$, the groups $\mathrm{Jac}_C(\mathbb{F}_q)$ and $\mathrm{Pic}_C^0(\mathbb{F}_q)$ are isomorphic. For our purposes, it will suffice to work with the Picard group.

Definition 2.6. Let $r \geq 0$. A divisor $D = P_1 + \ldots + P_r - r\mathcal{O} \in \mathrm{Div}_C$ is called *semi-reduced* if

- $P_i \in C \setminus \{\mathcal{O}\}$ and
- $P_i \neq w(P_j)$ for $i \neq j$ (while $P_i = P_j$ is possible).

If in addition $r \leq g$, then D is called *reduced*.

It follows from the Riemann–Roch Theorem (see [Sti93, Theorem I.5.15]) that every divisor class can be represented by a unique reduced divisor. Notice that $r = 0$ if and only if $[D] = 0$. We have $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ if and only if $\varphi^n(\{P_1, \ldots, P_r\}) = \{P_1, \ldots, P_r\}$ as multisets. For any divisors D_1 and D_2, we denote by $D_1 \oplus D_2$ the reduced divisor such that $D_1 \oplus D_2 \sim D_1 + D_2$.

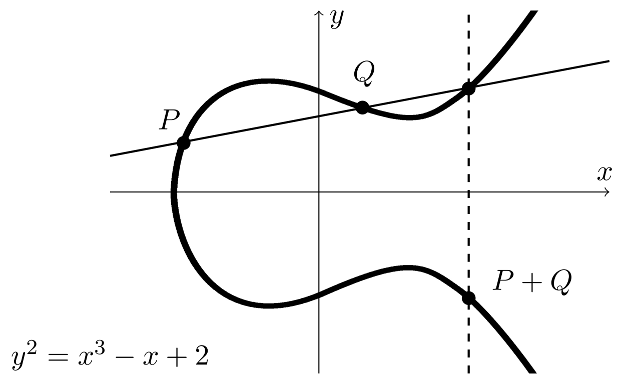
Remark 2.7. When $C = E$ is an elliptic curve, then each non-zero element of Pic_E^0 is uniquely represented by a divisor of the form $P - \mathcal{O}$ with $P \in E$. In fact,

$$P \mapsto [P - \mathcal{O}]$$

gives a bijection between E and Pic_E^0, and the points on E itself form a group. Therefore, E is isomorphic to its Jacobian variety and is itself an *abelian variety*. Whenever we deal with an elliptic curve, we work directly with the points on E. (One might say that we denote a divisor class by the unique $P \in E$ corresponding to $[P - \mathcal{O}] \in \mathrm{Pic}_E^0$, and in particular, that we denote $0 \in \mathrm{Pic}_E^0$ by the point $\mathcal{O}$—notice that this is not the reduced representation of the divisor zero).

We denote the group law on E induced by the group law on Pic_E^0 again by $\oplus$. It can be given by simple rational functions, see e.g. [ACD+06, Chapter 13.1.1] for the general formulas or [Was08, Chapter 2.2] for $\mathrm{char}\,\mathbb{F}_q \neq 2, 3$. For elliptic curves defined over the real numbers, this group law has a popular geometric interpretation known as the chord and tangent rule (see Figure 2.1). The neutral element of the group is the point at infinity

Figure 2.1 Chord and tangent rule for elliptic curve point addition (over $\mathbb{R}$)



$\mathcal{O}$. The inverse of an affine point $P = (X, Y)$ is $w(P) = \ominus P = (X, -Y)$. The Frobenius map on E is a group homomorphism.

We have $E(\mathbb{F}_{q^n}) \cong \mathrm{Pic}_E^0(\mathbb{F}_{q^n})$ for all $n \geq 1$. Elliptic curve cryptography usually uses $E(\mathbb{F}_{q^n})$, which is a finite subgroup of E of cardinality about q^n.

Another very useful representation of divisor classes is the following.

Definition 2.8. Let $[D] \in \mathrm{Pic}_C^0$ with $D = P_1 + \ldots + P_r - r\mathcal{O}$ a semi-reduced divisor and $P_i = (X_i, Y_i)$. The *Mumford representation* of $[D]$ is the pair of polynomials $[u(x), v(x)]$ where

- $u(x) = \prod_{i=1}^r (x - X_i)$
- $v(x)$ is such that $\deg v < \deg u$ and u divides $v^2 - f$.

There is a one-to-one correspondence between *reduced* divisors D and pairs $[u, v]$ such that u is monic, $\deg v < \deg u \leq g$, and $u \mid v^2 - f$: Given the divisor D, then $u(x) = \prod_{i=1}^r (x - X_i)$ and $v(x)$ is the unique polynomial such that $v(X_i) = Y_i$ with multiplicity equal to the multiplicity of P_i in D. The polynomial $v(x)$ may be computed by solving a linear system. Conversely, given the Mumford representation $[u, v]$, the corresponding reduced divisor is the D with defining ideal $I_D = (u(x), y - v(x))$. Notice that the curve equation $y^2 - f$ becomes superfluous, since $y^2 - f \in (u, y - v)$.

A convenient property of the Mumford representation is that $\mathbb{F}_{q^n}$-rationality of divisor classes becomes easily visible: We have $[u, v] \in \mathrm{Pic}_C^0(\mathbb{F}_{q^n})$ if and only if $u, v \in \mathbb{F}_{q^n}[x]$. It follows from the definition that the Mumford representation of $[0]$ is $[1, 0]$.

Remark 2.9. If $C = E$ is an elliptic curve, then the Mumford representation of $P = (X, Y) \in E$ is $[x - X, Y]$.

Definition 2.10. Let $\mathbb{F}|\mathbb{F}_q$ be an algebraic field extension. A divisor $D \in \mathrm{Div}_C(\mathbb{F})$ with Mumford representation $[u, v]$ is *prime* if $u \in \mathbb{F}[x]$ is an irreducible polynomial.

Notice that being prime depends on the choice of $\mathbb{F}$. When $\mathbb{F} = \mathbb{F}_{q^n}, n \geq 1$, it will sometimes be convenient to write a divisor as a sum of prime divisors: $D = D_1 + \ldots + D_t$, $D_i \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ prime. Notice that we may have $D_i = D_j$ for $i \neq j$ and $D_1, \ldots, D_t$ are unique up to permutation.

The Mumford representation is particularly useful when computing with divisor classes of hyperelliptic curves, and all algorithms given in Chapter 5 make use of this representation. *Cantor's Algorithm* [Can87] performs the addition of divisor classes in the Mumford representation. An easy modification computes not only $D_1 \oplus D_2$, but also a function a such that $D_1 + D_2 = D_1 \oplus D_2 + \operatorname{div}(a)$. For future reference, we give this algorithm in Algorithm 2.1. For elliptic curves, the standard addition formulas are easier to use and more efficient than Cantor's Algorithm. Also for genus 2 curves, there exist explicit addition formulas that are more efficient than Cantor's Algorithm, see [LS04, Lan05].

Algorithm 2.1 Cantor's Algorithm including rational function

```
Input: $[u_1, v_1], [u_2, v_2] \in \operatorname{Pic}^0_C$ in Mumford representation
Output: $[u, v]$ in Mumford representation, $a$ such that $[u, v] + \operatorname{div}(a) = [u_1, v_1] + [u_2, v_2]$
1: $a \leftarrow \gcd(u_1, u_2, v_1 + v_2)$, find $e_1, e_2, e_3$ such that $a = e_1u_1 + e_2u_2 + e_3(v_1 + v_2)$
2: $u \leftarrow u_1u_2/a^2$
3: $v \leftarrow (u_1v_2e_1 + u_2v_1e_2 + (v_1v_2 + f)e_3)/a \bmod u$
4: while $\deg u > g$ do
5:     $\tilde{u} \leftarrow \operatorname{monic}((f - v^2)/u), \tilde{v} \leftarrow -v \bmod \tilde{u}$
6:     $a \leftarrow a \cdot (y - v)/\tilde{u}$
7:     $u \leftarrow \tilde{u}, v \leftarrow \tilde{v}$
8: end while
9: return $[u, v], a$
```

The most frequent operation in the context of DLP-based cryptography is *scalar multiplication*, i.e. the computation of $k[D] = [kD]$ from $[D] \in \operatorname{Pic}^0_C$ for some $k \geq 1$. When working with an elliptic curve, we use the notation

$$[k]P = \underbrace{P \oplus P \oplus \ldots \oplus P}_{k-\text{times}}.$$

The standard way to perform scalar multiplication is with a *double and add* algorithm, as given e.g. in [ACD+06, Algorithm 13.6]. It computes $k[D]$ via a sequence of doublings and additions in $\log_2 k$ steps. This procedure is analogous to the *square and multiply* algorithm for exponentiation.

A number of more efficient variants of the square-and-multiply and double-and-add algorithms exist. They include using the non-adjacent form (NAF) in order to reduce the number of additions (see [ACD+06, Chapter 9.1.4]) and sliding-window methods (see [ACD+06, Algorithm 9.10]). A particularly clever variant developed especially for elliptic curves is the *Montgomery ladder* [Mon87], which computes the x-coordinate of $[k]P$ directly from the x-coordinate of P. It provides built-in protection against side-channel attacks, since it carries out an addition in each step, regardless of the shape of the binary representation of k. It has been generalized to hyperelliptic curves of genus 2 [Duq04, Lan04a, Gau07].

Efficient elliptic and hyperelliptic curve arithmetic is, in fact, a large and active field of research, and we do not go into detail here. A good overview is given in [ACD+06, Chapter 13, 14]. We point out that the implementation of efficient (hyper)elliptic curve arithmetic always builds on an efficient implementation of finite field arithmetic (cf. [ACD+06, Chapter 11]).

Another important topic in (hyper)elliptic curve cryptography are *point counting algorithms*, which compute the order of $\operatorname{Pic}^0_C(\mathbb{F}_q)$. This usually amounts to computing the characteristic polynomial of the Frobenius endomorphism. The standard method for elliptic curves is *Schoof's Algorithm* [Sch85] and its variants, in particular the improvements due to Atkin and Elkies, resulting in the *SEA Algorithm*. It can handle groups of cryptographic size. For hyperelliptic curves, point counting is still a bottleneck today. The record for genus 2 curves over prime fields is held by Gaudry and Schost [GS12], using an extension of Schoof's algorithm. For more information on this topic, see [ACD+06, Chapter 17].

2.3 Weil restriction

Scalar restriction, first introduced by Weil [Wei58] and therefore often called *Weil restriction* or *Weil descent*, is a well-known technique in arithmetic algebraic geometry. It can be applied to many geometric objects, for example algebraic varieties. For a given finite separable field extension $L|K$ of degree n, it relates d-dimensional objects over L to nd-dimensional objects over K.

Frey [Fre99] first suggested to study Weil restriction of abelian varieties in the context of DLP-based cryptography, since the structure of the object after scalar restriction can be much richer than that of the original variety. He suggests both constructive and destructive applications of this concept, the constructive one being the use of the trace zero variety, which is a subvariety of the Weil restriction of an elliptic curve or the Jacobian variety associated to a hyperelliptic curve and the object of study of this thesis.

We give only a very superficial introduction to the topic, which will be sufficient for the content of this thesis. For a through treatment, we refer to [BLR80, Die01] and the references therein. For a description more dedicated to the purposes of cryptography, see [ACD+06, Chapter 7].

Definition 2.11. Let $L|K$ be a finite Galois extension of degree n, and let V be an affine (resp. projective) algebraic variety V defined over L of dimension d. The *Weil restriction* of V is an affine (resp. projective) algebraic variety $\operatorname{Res}_{L|K} V$ defined over K of dimension nd such that for every field extension $F|K$ there is a functorial bijection

$$(\operatorname{Res}_{L|K} V)(F) \cong V(F \otimes_K L). \tag{2.2}$$

More generally, $\operatorname{Res}_{L|K}$ is a functor fulfilling a certain universal property and mapping schemes to sets. Naturally, the Weil restriction of an object inherits many properties from the original object. In particular, it can be shown that the Weil restriction of an affine (resp. projective) variety always exists and is again an affine (resp. projective) variety.

Here we are interested in much more concrete matters concerning Weil restriction of varieties. Most importantly, there is a simple procedure for writing equations for the affine variety $\operatorname{Res}_{L|K} V$ from equations of the affine variety V. By a slight abuse of terminology, we use the term *Weil restriction* not only for the object obtained by this procedure (i.e. a variety), but also for the procedure itself, although some may prefer to call the latter Weil descent. In particular for us, *Weil restriction with respect to the extension* $L|K$ is a procedure that can be applied to a multivariate polynomial with coefficients in L and results in n polynomials with coefficients in K. In this process, the number of indeterminates is multiplied by n.

Definition 2.12. Let $L|K$ be a Galois extension of degree n, and let $\{\zeta_1, \ldots, \zeta_n\}$ be a basis of L as a K-vector space. For a polynomial $G \in L[x_1, \ldots, x_\ell]$, carry out the following steps:

- Define ℓn indeterminates x_{ik} by

$$x_i = x_{i1}\zeta_1 + x_{i2}\zeta_2 + \ldots + x_{in}\zeta_n \qquad \text{for } i = 1, \ldots, \ell.$$

- Replace the indeterminates x_i in $G(x_1, \dots, x_\ell)$ by the above expressions, which gives
$$G\left(\sum_{k=1}^{n} x_{1k}\zeta_k, \dots, \sum_{k=1}^{n} x_{\ell k}\zeta_k\right).$$
- Write the coefficients of G (which are in L) as K-linear combinations of the basis $\{\zeta_1, \dots, \zeta_n\}$ and sort the terms according to this basis, giving
$$\sum_{k=1}^{n} g_k(x_{11}, \dots, x_{\ell n})\zeta_k.$$

Then we call the polynomials $g_1, \dots, g_n \in K[x_{11}, \dots, x_{\ell n}]$ the *Weil restriction* of the polynomial G, and we refer to this procedure as *Weil restricting* G.

Theorem 2.13. *Let V be an affine variety defined over L by m equations*
$$G_1, \dots, G_m \in L[x_1, \dots, x_\ell],$$
i.e. we have
$$V = \{(X_1, \dots, X_\ell) \mid G_j(X_1, \dots, X_\ell) = 0 \textit{ for } j = 1, \dots, m\}.$$
For each $j = 1, \dots, m$, let $g_{j1}, \dots, g_{jn} \in K[x_{11}, \dots, x_{\ell n}]$ be the Weil restriction of G_j. Then we have
$$\operatorname{Res}_{L|K} V = \{(X_{11}, \dots, X_{\ell n}) \mid g_{jk}(X_{11}, \dots, X_{\ell n}) = 0 \textit{ for } j = 1, \dots, m, k = 1, \dots, n\},$$
i.e. the variety $\operatorname{Res}_{L|K} V$ is defined over K by the mn polynomials $g_{jk} \in K[x_{11}, \dots, x_{\ell n}]$.

This construction makes explicit the most important special case of property (2.2): For $F = K$, we get
$$(\operatorname{Res}_{L|K} V)(K) \cong V(L),$$
i.e. the K-points of $\operatorname{Res}_{L|K} V$ correspond to the L-points of V via the map
$$(X_{11}, \dots, X_{\ell n}) \longleftrightarrow \left(\sum_{k=1}^{n} X_{1k}\zeta_k, \dots, \sum_{k=1}^{n} X_{\ell k}\zeta_k\right).$$

The Weil restriction process explained above can also be applied to projective varieties, since any projective variety V can be covered by affine subvarieties V_i. By Weil restricting the V_i, one gets a collection of affine varieties $W_i := \operatorname{Res}_{L|K} V_i$. When a suitable covering is chosen and the W_i are glued together in projective space in a suitable way (see [ACD+06, Chapter 7.2]), one obtains a projective variety $W = \operatorname{Res}_{L|K} V$.

It is also intuitive (and can, of course, be proven rigorously) that the Weil restriction of an abelian variety is again an abelian variety. This is because the group law of the original variety, defined by rational functions, naturally carries over into the Weil restriction. Its defining rational functions can be found by Weil restricting the original rational functions. Therefore, the Weil restriction of both elliptic curves and Jacobians of hyperelliptic curves are again abelian varieties.

In cryptography, where one uses elliptic curves and Jacobians of hyperelliptic curves defined over finite fields $\mathbb{F}_{q^n}$, Weil restriction with respect to the extension $\mathbb{F}_{q^n}|\mathbb{F}_q$ yields associated abelian varieties defined over $\mathbb{F}_q$. It is the study of these varieties that Frey suggests in [Fre99] for constructive and destructive applications. For example, when E is an elliptic curve defined over $\mathbb{F}_{q^n}$ for some $n \geq 1$, then its Weil restriction W_E is an n-dimensional variety that satisfies

$$W_E(\mathbb{F}_q) \cong E(\mathbb{F}_{q^n}).$$

We study an example of this in Example 2.14. More generally, for a hyperelliptic curve C of genus g, we have the Weil restriction W_C of the Jacobian variety of C, which has dimension ng and satisfies

$$W_C(\mathbb{F}_q) \cong \mathrm{Pic}^0_C(\mathbb{F}_{q^n}).$$

In both cases, the functorial bijection gives a group isomorphism, and the group law translates naturally from the group on the right to the one on the left. The trace zero variety, which we introduce in the next section, is a subvariety of W_E resp. W_C, and its $\mathbb{F}_q$-rational points yield a subgroup of $E(\mathbb{F}_{q^n})$ resp. $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$.

Example 2.14. We apply Weil restriction to the affine part of an elliptic curve given by the equation

$$y^2 - x^3 - Ax - B = 0.$$

For simplicity and because this case will be relevant later on, we assume that E is defined over $\mathbb{F}_q$, where $3 \mid q-1$. We perform Weil descent with respect to a Kummer extension $\mathbb{F}_{q^3}|\mathbb{F}_q$, where $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - \mu)$ for some $\mu \in \mathbb{F}_q$ that is not a third power. Then $\{1, \zeta, \zeta^2\}$ is a basis of $\mathbb{F}_{q^3}|\mathbb{F}_q$. We write

$$\begin{aligned} x &= x_0 + x_1\zeta + x_2\zeta^2 \\ y &= y_0 + y_1\zeta + y_2\zeta^2 \end{aligned}$$

and plug this into the curve equation, which gives

$$(y_0 + y_1\zeta + y_2\zeta^2)^2 - (x_0 + x_1\zeta + x_2\zeta^2)^3 - A(x_0 + x_1\zeta + x_2\zeta^2) - B = 0.$$

Multiplying this out, substituting $\zeta^3 = \mu$, and collecting the coefficients of $1, \zeta, \zeta^2$, we get

$$\begin{aligned} y_0^2 + 2\mu y_1 y_2 - x_0^3 - \mu x_1^3 - \mu^2 x_2^3 - 6\mu x_0 x_1 x_2 - Ax_0 - B &= 0 \\ 2y_0y_1 + \mu y_2^2 - 3x_0^2 x_1 - 3\mu x_0 x_2^2 - 3\mu x_1^2 x_2 - Ax_1 &= 0 \\ 2y_0y_2 + y_1^2 - 3x_0^2x_2 - 3x_0x_1^2 - 3\mu x_1 x_2^2 - Ax_2 &= 0. \end{aligned}$$

So from one equation in two indeterminates x, y, which describes a one-dimensional variety in affine 2-space, we have produced three equations in six variables $x_0, x_1, x_2, y_0, y_1, y_2$, which describe a three-dimensional variety in affine 6-space. If W_E is the projective Weil restriction of the projective variety E, then these equations describe the affine part of W_E. We have

$$\begin{aligned} W_E(\mathbb{F}_q) &\cong E(\mathbb{F}_{q^n}) \\ (X_0, X_1, X_2, Y_0, Y_1, Y_2) &\leftrightarrow (X, Y) = (X_0 + X_1\zeta + X_2\zeta^2, Y_0 + Y_1\zeta + Y_2\zeta^2). \end{aligned}$$

The group law in W_E can be defined in terms of the coordinates $x_0, x_1, x_2, y_0, y_1, y_2$.

2.4 The trace zero variety

The trace zero variety associated to an elliptic or hyperelliptic curve was first proposed in the context of cryptography by Frey [Fre99] and further studied by Naumann [Nau99], Weimerskirch [Wei01], Blady [Bla02], Lange [Lan01, Lan04b], Diem and Scholten [Die01, DS03, DS], Rubin and Silverberg [Sil05, RS09], and Avanzi and Cesena [AC07, Ces08, Ces10]. It gives rise to subgroups of the (Picard group of the) associated curve that are interesting in the context of DLP-based cryptography. We will explain this after giving the definition and some basic facts.

Throughout this section, we let C be an elliptic or hyperelliptic curve defined over a finite field $\mathbb{F}_q$.

Definition 2.15. The *trace endomorphism* in the divisor group of C with respect to the extension $\mathbb{F}_{q^n}|\mathbb{F}_q, n \geq 1$, is defined by

$$\mathrm{Tr} : \mathrm{Div}_C(\mathbb{F}_{q^n}) \to \mathrm{Div}_C(\mathbb{F}_q), \quad D \mapsto D + \varphi(D) + \ldots + \varphi^{n-1}(D).$$

The following properties of the trace will be useful in Chapter 5.

Lemma 2.16. *The trace homomorphism* $\mathrm{Tr} : \mathrm{Div}_C(\mathbb{F}_{q^n}) \to \mathrm{Div}_C(\mathbb{F}_q)$ *has the following properties:*

(*i*) *For any prime divisor D we have* $\mathrm{Tr}^{-1}(\mathrm{Tr}(D)) = \{D, \varphi(D), \ldots, \varphi^{n-1}(D)\}$.

(*ii*) $D \in \mathrm{Div}_C(\mathbb{F}_{q^n}) \setminus \mathrm{Div}_C(\mathbb{F}_q)$ *is a prime divisor if and only if* $\mathrm{Tr}(D) \in \mathrm{Div}_C(\mathbb{F}_q)$ *is a prime divisor.*

Proof. We denote by u^φ the application of the finite field Frobenius automorphism $\varphi : \overline{\mathbb{F}}_q \to \overline{\mathbb{F}}_q$ to the coefficients of a polynomial u. We denote the product $uu^\varphi \cdots u^{\varphi^{n-1}}$ by $u^{1+\varphi+\ldots+\varphi^{n-1}}$.

(*i*) Let $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ be given with u-polynomial $u \in \mathbb{F}_{q^n}[x]$ irreducible. Then $\mathrm{Tr}(D)$ has u-polynomial $N(u) = uu^\varphi \cdots u^{\varphi^{n-1}}$, where all the u^{φ^j} are irreducible over $\mathbb{F}_{q^n}$. Hence any D' with $\mathrm{Tr}(D') = \mathrm{Tr}(D)$ has to have as u-polynomial one of the u^{φ^j}, and therefore $D' = \varphi^j(D)$ for some $j \in \{0, \ldots, n-1\}$. Conversely, $\mathrm{Tr}(\varphi^j(D)) = \mathrm{Tr}(D)$ for all j.

(*ii*) We need to show that $u \in \mathbb{F}_{q^n}[x]$ is irreducible if and only if $N(u) = uu^\varphi \cdots u^{\varphi^{n-1}} \in \mathbb{F}_q[x]$ is irreducible. So first suppose that $u = u_1u_2$ with $u_1, u_2 \in \mathbb{F}_{q^n}[x]$. Then $N(u) = N(u_1u_2) = N(u_1)N(u_2)$ with $N(u_1), N(u_2) \in \mathbb{F}_q[x]$. Conversely, suppose that $N(u) = U_1U_2$ with $U_1, U_2 \in \mathbb{F}_q[x] \setminus \mathbb{F}_q$. Now since $N(u) = uu^\varphi \cdots u^{\varphi^{n-1}}$ is the factorization of $N(u)$ into irreducible polynomials over $\mathbb{F}_{q^n}$, we have

$$U_i = \prod_{j \in S_i} u^{\varphi^j}$$

for $i = 1, 2$ and $S_1 \dot{\cup} S_2 = \{0, \ldots, n-1\}$, $S_1, S_2 \neq \emptyset$. But $U_i \in \mathbb{F}_q[x]$ implies $U_i^\varphi = U_i$, which yields a contradiction unless $u \in \mathbb{F}_q[x]$. □

Since the Frobenius map is well-defined as an endomorphism on divisor classes, we also have a trace endomorphism [Tr] in the Picard group

$$[\mathrm{Tr}] : \mathrm{Pic}^0_C(\mathbb{F}_{q^n}) \to \mathrm{Pic}^0_C(\mathbb{F}_q), \quad [D] \mapsto [D + \varphi(D) + \ldots + \varphi^{n-1}(D)].$$

We are interested in the kernel of this map.

Definition 2.17. Let n be a prime number. Then the *trace zero subgroup* of $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ is

$$T_n = \{[D] \in \mathrm{Pic}^0_C(\mathbb{F}_{q^n}) \mid \mathrm{Tr}(D) \sim 0\}.$$

Theorem 2.18. *The points of T_n can be viewed as the $\mathbb{F}_q$-rational points of the* trace zero variety. *This is an $(n-1)g$-dimensional subvariety of the Weil restriction with respect to $\mathbb{F}_{q^n}|\mathbb{F}_q$ of the Jacobian variety of C. The trace zero variety is therefore defined over $\mathbb{F}_q$. Throughout this work, we denote it by V_n.*

For a proof and more details, see [ACD+06, Chapters 7.4.2 and 15.3]. We explain how to write equations for the trace zero variety in Chapter 3. From Theorem 2.13, we obtain a natural correspondence between the points of T_n and the points of $V_n(\mathbb{F}_q)$. Namely, for a fixed polynomial basis $\{1, \zeta \ldots, \zeta^{n-1}\}$ of $\mathbb{F}_{q^n}|\mathbb{F}_q$ we have

$$\begin{array}{rcl} V_n(\mathbb{F}_q) & \cong & T_n \\ (X_0, \ldots, X_{n-1}, Y_0, \ldots, Y_{n-1}) & \leftrightarrow & (X, Y) = (X_0 + \ldots + X_{n-1}\zeta^{n-1}, Y_0 + \ldots + Y_{n-1}\zeta^{n-1}). \end{array}$$

We often identify $T_n = V_n(\mathbb{F}_q)$ via this correspondence without mentioning it. Since V_n has dimension $(n-1)g$, we have that $|T_n| = |V_n(\mathbb{F}_q)| \approx q^{(n-1)g}$ by [LW54].

As mentioned before, interest in the trace zero variety in the cryptographic context was first raised by Frey in [Fre99]. One of the motivations for using the trace zero subgroup in cryptosystems is that it allows in principle to reduce the key length. In Chapter 4, we show how to do this for elliptic curves and small n, and in Chapters 5 and 6, we show how to do this for elliptic and hyperelliptic curves of any genus g and with respect to any prime n.

Moreover, addition in the trace zero subgroup may be sped up considerably by using the Frobenius endomorphism. The idea is that in a cyclic group, φ operates as multiplication by some number s, which may be computed from the characteristic polynomial of φ. If

$$k = k_0 + k_1 s + \ldots + k_t s^t$$

is the base-s expansion of k, then $k[D]$ may be computed as

$$k[D] = k_0[D] + k_1\varphi([D]) + \ldots + k_t\varphi^t([D]).$$

This is more efficient than computing $k[D]$ with double-and-add directly, since applying φ to $[D]$ is very cheap. This method was studied in detail by Lange (see [Lan01, Lan04b]) and by Avanzi and Cesena (see [AC07, Ces08]). The approach is similar to the one used for Koblitz curves (see [Kob91]) and was later applied to GLV/GLS curves (see [GLV01, GLS11]).

Furthermore, Rubin and Silverberg showed in [RS09] that pairings defined on higher dimensional abelian varieties, such as the trace zero variety, yield exceptionally high security parameters for some values of n and g.

Finally, the trace zero subgroup captures the hardness of the DLP over extension fields. More specifically, the DLP in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ is as hard as the DLP in T_n when C is defined over $\mathbb{F}_q$.

Proposition 2.19. *We have a short exact sequence*

$$0 \longrightarrow \mathrm{Pic}^0_C(\mathbb{F}_q) \longrightarrow \mathrm{Pic}^0_C(\mathbb{F}_{q^n}) \xrightarrow{[\varphi - \mathrm{id}]} T_n \longrightarrow 0.$$

In particular, solving a DLP in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ has the same complexity as solving a DLP in T_n and a DLP in $\mathrm{Pic}^0_C(\mathbb{F}_q)$.

Proof. Surjectivity of $[\varphi - \mathrm{id}]$ holds according to [ACD+06, Proposition 7.13]. This proves that we have a short exact sequence as claimed.

As explained in [GV05] for the analogous case of algebraic tori, a DLP in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ may be mapped to a DLP in $\mathrm{Pic}^0_C(\mathbb{F}_q)$ by the trace map, and it may be solved in $\mathrm{Pic}^0_C(\mathbb{F}_q)$ modulo the order of that group. The remaining modular information required to compute a discrete logarithm in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ comes from solving a DLP in T_n. A formal argument stating that solving a DLP in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ has the same complexity as solving a DLP in T_n and solving a DLP in $\mathrm{Pic}^0_C(\mathbb{F}_q)$ is given by Galbraith and Smith [GS06]. $\square$

As a consequence of the exact sequence in Proposition 2.19 we obtain

$$|T_n| = \frac{|\mathrm{Pic}^0_C(\mathbb{F}_{q^n})|}{|\mathrm{Pic}^0_C(\mathbb{F}_q)|}.$$

The Hasse–Weil Theorem states that

$$|\mathrm{Pic}^0_C(\mathbb{F}_{q^n})| = \prod_{i=1}^{2g}(1-\tau_i^n),$$

where τ_i are the roots of the characteristic polynomial of φ. This may be used to give simple formulas for the cardinality of the trace zero subgroup in terms of the coefficients of the characteristic polynomial, see [ACD+06, Chapter 15.3.1]. This shows another advantage of using trace zero subgroups in the cryptographic setting, where it is essential to work with a group of known prime or almost prime order: Counting the number of points in T_n only requires determining the characteristic polynomial of a curve defined over $\mathbb{F}_q$. Counting the number of points of an elliptic or hyperelliptic curve of, e.g., the same genus and comparable group size would require determining the characteristic polynomial of a curve defined over $\mathbb{F}_{q^{n-1}}$.

Remark 2.20. The choice of good parameters is crucial for the security of trace zero cryptosystems. While Lange [Lan04b], Avanzi and Cesena [AC07], and Rubin and Silverberg [RS09] have shown that for certain choices of n and g trace zero subgroups are useful and secure in the context of pairing-based cryptography, there may be security issues in connection with DLP-based cryptosystems for some parameters. See Chapter 7.5 for a discussion.

2.5 Optimal representation

One of the goals of this work is finding an optimal-size representation for the elements of the trace zero subgroup. This is a natural question, which has been investigated in previous works both for elliptic and hyperelliptic curves [Nau99, Lan01, Lan04b, Sil05], as well as in the analogous case of torus-based cryptography [RS03]. It is also stated as an open problem in the conclusions of [AC07] and in [RS09]. For further discussion on the significance of compact representations, see [Gor11].

Definition 2.21. A *representation of size ℓ* for the elements of a finite set G is an injection

$$\mathcal{R} : G \longrightarrow \mathbb{F}_2^\ell.$$

A representation $\mathcal{R}$ is *optimal* if it is of size $\log_2|G| + O(1)$. Given $\gamma \in G$ and $x \in \mathrm{Im}\,\mathcal{R}$, we refer to computing $\mathcal{R}(\gamma)$ as *compression* and $\mathcal{R}^{-1}(x)$ as *decompression*.

Abusing terminology, in this thesis we call representation a map $\mathcal{R}$ with the property that an element of $\mathbb{F}_2^\ell$ has at most d inverse images, for some small fixed d. In this case, we say that $x \in \operatorname{Im} \mathcal{R}$ is a representation for the *class* $\mathcal{R}^{-1}(x)$. Notice that the number of classes is about $|G|/d \approx |G|$, if d is a small constant. Moreover, we call $\mathcal{R}$ optimal if $\ell \approx \log_2 |G|$, or equivalently $|G| \approx 2^\ell$, since intuitively the representation is optimal if its size is *close to* $\log_2 |G|$.

Remark 2.22. Since the elements of $\mathbb{F}_q$ can be represented via binary strings of length $\log_2 q$, an optimal representation for a set G with $|G| \approx q^m$ can be given via $\mathcal{R} : G \longrightarrow \mathbb{F}_q^m$.

Example 2.23. Consider the usual representation for points on an elliptic curve E defined over $\mathbb{F}_q$

$$\begin{array}{rcl} \mathcal{R} : E(\mathbb{F}_q) \setminus \{\mathcal{O}\} & \longrightarrow & \mathbb{F}_q \\ (X, Y) & \longmapsto & X. \end{array}$$

Compression has no computational cost, and decompression is efficient, since Y can be recomputed, up to sign, from the equation of the curve at the cost of computing a square root in $\mathbb{F}_q$. The representation is optimal, since $|E(\mathbb{F}_q)| \approx q$ by Hasse's Theorem.

For any $X \in \mathcal{R}(E(\mathbb{F}_q))$ we have $\mathcal{R}^{-1}(X) = \{(X,Y),(X,-Y)\}$, hence the representation identifies each point with its negative. We can therefore think of working with classes of size two. This is compatible with scalar multiplication, since $-k(X,Y) = k(X,-Y)$ for all $(X,Y) \in E$ and for all $k \in \mathbb{N}$.

A simple way to make the above representation bijective is to append to the image of each point an extra bit corresponding to the sign of the y-coordinate. This gives a representation

$$\mathcal{R}' : E(\mathbb{F}_q) \setminus \{0\} \longrightarrow \mathbb{F}_q \times \mathbb{F}_2$$

of size $\log_2 q + 1$, which is optimal for large q, since $\log_2 |E(\mathbb{F}_q)| \approx \log_2 q \approx \log_2 q + 1$.

In fact, we cannot expect to find a bijective representation smaller than $\log_2 q + 1$ bits for general elliptic curves since by Hasse's bound, which is actually attained for most fields (see [Wat69]), $E(\mathbb{F}_q)$ can have up to $\lfloor q + 2\sqrt{q} + 1 \rfloor$ elements, and $\log_2 q < \log_2(q + 2\sqrt{q} + 1) \leq \log_2 q + 1$ for sufficiently large q.

The above logic can also be applied to hyperelliptic curves.

Example 2.24. Let C be a hyperelliptic curve of genus g defined over $\mathbb{F}_q$. Using the u-polynomial $u(x) \in \mathbb{F}_q[x]$ of the Mumford representation to represent a point $[u, v] \in \operatorname{Pic}_C^0(\mathbb{F}_q)$ gives an optimal representation. In fact, u has degree $r \leq g$, and $|\operatorname{Pic}_C^0(\mathbb{F}_q)| \approx q^g$ by the Theorem of Hasse–Weil. Intuitively, since u is monic, we do not need to store its leading coefficient, but rather at most g coefficients and the degree of u. Concretely, since $\deg u = g$ for most divisors, and in order to have representations all of the same length, we choose to represent u via the coefficients of $x^{g-1}, \ldots, 1$, plus an extra bit δ which is 1 if the degree of u is g and 0 otherwise. More precisely, we let

$$\begin{array}{rcl} \mathcal{R} : \operatorname{Pic}_C^0(\mathbb{F}_q) & \longrightarrow & \mathbb{F}_q^g \times \mathbb{F}_2 \\ {[u = \sum_{i=0}^g u_i x^i, v]} & \longmapsto & (u_0, \ldots, u_{g-1}, \delta) \end{array}$$

where $u_i = 0$ for $i > r = \deg u$, $\delta = 1$ if $r = g$, and 0 otherwise.

The polynomial u contains all the information about the x-coordinates of the points P_i in the reduced representation of $D = P_1 + \ldots + P_r - r\mathcal{O}$, but not about the signs of the corresponding y-coordinates. As before, one can either use g extra bits to store these signs (see Hess, Seroussi, and Smart [HSS01]), or one can work with classes $\{[w^{i_1}(P_1) + w^{i_2}(P_2) + \ldots + w^{i_r}(P_r) - r\mathcal{O}] \mid i_j \in \{0,1\}\}$, thus identifying up to 2^g elements of $\operatorname{Pic}_C^0(\mathbb{F}_q)$.

For small g, these classes are not too large. A different representation for the elements of $\mathrm{Pic}^0_C(\mathbb{F}_q)$ of size $g\log_2 q + g$ is given by Stahlke [Sta04].

Remark 2.25. Since $T_n \subseteq \mathrm{Pic}^0_C(\mathbb{F}_{q^n})$, we may use the representation of Example 2.24 for points of the trace zero subgroup. However this is not optimal, since $\log_2 |T_n| \approx (n-1)g\log_2 q \not\approx ng\log_2 q$. In Chapters 4, 5, and 6, we study optimal representations for the elements of T_n.

2.6 Gröbner bases

A Gröbner basis is a special generating set of an ideal in a multivariate polynomial ring. It is a powerful tool when studying such ideals in computational algebraic geometry and commutative algebra, and one important application is that a Gröbner basis can be used to solve a system of multivariate polynomial equations. The concept was introduced by Buchberger [Buc65] (and independently by several others, see [Eis04, Chapter 15.6] for further names and references), and standard references on the topic are [CLO92, KR00, KR05].

Let $F[x] = F[x_1,\ldots,x_m]$ be a polynomial ring in m variables over any field F. For $\alpha = (\alpha_1,\ldots,\alpha_m) \in \mathbb{Z}^m_{\geq 0}$, we write *monomials* as $x^\alpha = x_1^{\alpha_1}\cdot\ldots\cdot x_m^{\alpha_m}$. Then cx^α for some $c \in F$ and some exponent vector α is a *term.* The *(total) degree* of a monomial x^α is $\deg(x^\alpha) = \sum_{i=1}^m \alpha_i$, and its *multidegree* is α.

Definition 2.26. A *monomial order* or *term order* on $F[x]$ is a relation $>$ on $\mathbb{Z}^m_{\geq 0}$, or equivalently on the set $\{x^\alpha \mid \alpha \in \mathbb{Z}_{\geq 0}\}$, satisfying

(i) $>$ is a total order on $\mathbb{Z}^m_{\geq 0}$,

(ii) if $\alpha > \beta$ and $\gamma \in \mathbb{Z}^m_{\geq 0}$, then $\alpha+\gamma > \beta+\gamma$, and

(iii) $\alpha \geq 0$ for all $\alpha \in \mathbb{Z}^m_{\geq 0}$.

Examples 2.27. Some common term orders are the following. Let $\alpha = (\alpha_1,\ldots,\alpha_m)$, $\beta = (\beta_1,\ldots,\beta_m) \in \mathbb{Z}^m_{\geq 0}$.

(i) The *lexicographic order* is defined by: $x^\alpha > x^\beta$ if the first non-zero entry in the vector difference $\alpha - \beta \in \mathbb{Z}^m$ is positive.

(ii) The *degree reverse lexicographic order* is defined by: $x^\alpha > x^\beta$ if $\deg(x^\alpha) > \deg(x^\beta)$, or if $\deg(x^\alpha) = \deg(x^\beta)$ and the last non-zero entry in the vector difference $\alpha - \beta \in \mathbb{Z}^m$ is negative. Because of the first condition, this is often called a *degree-compatible* order.

(iii) For $1 \leq j < m$, the *j-th elimination order* is defined by: $x^\alpha > x^\beta$ if $\alpha_1 + \ldots + \alpha_j > \beta_1 + \ldots + \beta_j$, or if $\alpha_1 + \ldots + \alpha_j = \beta_1 + \ldots + \beta_j$ and $x^\alpha > x^\beta$ with respect to the degree reverse lexicographic term order.

By attaching a term order $>$ to a multivariate polynomial ring, one may define the *leading monomial* of a polynomial, which is the monomial of maximum multidegree with respect to $>$. The *leading coefficient* is the coefficient of the leading monomial, and the *leading term* is the product of the leading coefficient and the leading monomial.

For the following definitions and statements, let us fix a term order $>$ on $F[x]$. For $f \in F[x]$, we write $\mathrm{lt}(f)$ for the leading term of f with respect to $>$.

Definition 2.28. Let $I \neq \{0\}$ be an ideal in $F[x]$. The ideal

$$\mathrm{lt}(I) = \langle \mathrm{lt}(f) \mid f \in I \rangle$$

generated by the leading terms of the elements of I is called the *leading term ideal*, or *initial ideal*, of I.

According to the Hilbert Basis Theorem, every ideal $I \subseteq F[x]$ has a finite generating set. If $I = \langle f_1, \ldots, f_s \rangle$, then it is always true that

$$\langle \mathrm{lt}(f_1), \ldots, \mathrm{lt}(f_s) \rangle \subseteq \mathrm{lt}(I),$$

but the initial ideal can be strictly larger.

Definition 2.29. A *Gröbner basis* of $\{0\} \neq I \subseteq F[x]$ is a finite subset $G = \{g_1, \ldots, g_t\}$ of I such that

$$\langle \mathrm{lt}(g_1), \ldots, \mathrm{lt}(g_t) \rangle = \mathrm{lt}(I).$$

It follows from the Hilbert Basis Theorem that a Gröbner basis exists for any non-zero ideal and $I = \langle g_1, \ldots, g_t \rangle$. A Gröbner basis is not unique, but uniqueness can be achieved by requiring additional properties, namely that all elements of G have leading coefficient 1 and that no monomial of f_i lies in $\mathrm{lt}(G \setminus \{f_i\})$ for all $i = 1, \ldots, t$. This is called a *reduced* Gröbner basis.

It is essential that a Gröbner basis is always defined with respect to a certain term order. In fact, the term order crucially influences its properties. In the following, we briefly explain two useful properties of Gröbner bases with respect to lexicographic and elimination term orders, respectively.

Theorem 2.30 (Shape Lemma, [KR00, Theorem 3.7.25])**.** *Let F be a perfect field. Let $I \subseteq F[x]$ be a zero-dimensional radical ideal, where any two zeros $(a_1, \ldots, a_m), (b_1, \ldots, b_m) \in \bar{F}^m$ of I satisfy $a_m \neq b_m$. Then the reduced Gröbner basis of I with respect to the lexicographic term order is of the form*

$$G = \{x_1 - g_1(x_m), \ldots, x_{m-1} - g_{m-1}(x_m), g_m(x_m)\}.$$

Moreover, if $d = \deg(g_m)$, then g_m has d distinct zeros $c_1, \ldots, c_d \in \bar{F}$, and the set of zeros of I is

$$Z(I) = \{(g_1(c_i), \ldots, g_{m-1}(c_i), c_i) \mid i = 1, \ldots, d\}.$$

This lemma shows how to solve a system

$$f_i(x_1, \ldots, x_m) = 0, \quad i = 1, \ldots, s$$

of $s \geq m$ polynomial equations $f_i \in F[x]$ in m indeterminates, assuming that there are only finitely many solutions over $\bar{F}$. Then the f_i generate a zero-dimensional ideal I, and if the technical conditions of the Shape Lemma are satisfied, then it guarantees that the lexicographic Gröbner basis of I is triangular, and hence the solutions of the system may easily be deduced after factoring g_m. In practice, this works most of the time and there is no need to verify the technical conditions.

Theorem 2.31 (Elimination Theorem, [KR00, Theorem 3.4.5])**.** *Let $1 \leq j < m$. If I is an ideal in $F[x]$ and G a Gröbner basis of I with respect to the j-th elimination order, then $G \cap F[x_{j+1}, \ldots, x_m]$ is a Gröbner basis of the j-th elimination ideal $I \cap F[x_{j+1}, \ldots, x_m]$.*

This means that Gröbner bases are useful for eliminating indeterminates from a polynomial system: In order to eliminate $x_1, \ldots, x_j$ from the system

$$f_i(x_1, \ldots, x_m) = 0, \quad i = 1, \ldots, s$$

of polynomial equations, we find a Gröbner basis of the ideal $\langle f_1, \ldots, f_s \rangle$ with respect to the j-th elimination order. The result gives a system only in the indeterminates $x_{j+1}, \ldots, x_m$, i.e. where the first j indeterminates have been eliminated.

An important question that remains is how to obtain a Gröbner basis of a given ideal. A number of algorithms have been proposed for this purpose, including Buchberger's algorithm [Buc65], Faugère's F4 and F5 algorithms [Fau99, Fau02], and the XL algorithm [CKPS00], which was originally introduced to solve overdetermined systems of polynomial equations and later shown to be a Gröbner basis algorithm similar to F4 (see [AFI$^+$04]). Buchberger's algorithm and F4 are most commonly used, and they are implemented in many computer algebra systems, including CoCoA [CoC], Macaulay2 [GS], Singular [DGPS12], Magma [BCP97], and Maple [Map].

Computing a Gröbner basis with such algorithms is often very costly. It was shown by Mayr and Meyer [MM82] that computing a Gröbner basis can take doubly exponential time in the degrees of the equations. While examples that have this property are specially constructed and problem instances that arise naturally are usually less costly, it is very difficult to give precise complexity bounds, see [GvzG99, Chapter 21.7] for a discussion.

The *Castelnuovo–Mumford regularity* gives an indication of the complexity of a Gröbner basis computation for a specific ideal. It is an invariant m_{reg} associated to any homogeneous multivariate ideal I, for a definition see [Eis04, Chapter 20.5]. Bayer and Stillman [BS88] proved that, after a generic change of coordinates, the polynomials in a reduced Gröbner basis with respect to the degree reverse lexicographic term order have degree at most m_{reg}. This means also that the degree of all polynomials that arise during the execution of Buchberger's algorithm is bounded by m_{reg}.

A closely related concept is the *regularity index* or *degree of regularity* of a homogeneous ideal, which roughly speaking is the smallest index d_{reg} where the Hilbert function agrees with the Hilbert polynomial, see [KR05, Definition 5.4.11]. This number is in general at most as large as the Castelnuovo–Mumford regularity, and for zero-dimensional ideals the two values are the same. There exists an explicit bound on the complexity of the F5 algorithm (for computing a degree reverse lexicographic Gröbner basis) which depends on the degree of regularity of the input ideal [BFP08, Proposition 2.2], and it can be improved assuming that the input to the algorithm is a *semi-regular sequence* (see [BFS04, BFSY05] for the definition of semi-regular and the bound). Since F5 has the same asymptotics as the XL algorithm, bounds on XL also apply, see [YC04a, YC04b, YCC04, YCY13]. However, computing d_{reg} is usually not easier than computing a Gröbner basis itself, and finding bounds on d_{reg} is difficult.

A way to bound d_{reg} is to use standard bounds on m_{reg} from commutative algebra, such as

$$d_{\mathrm{reg}} \le m_{\mathrm{reg}} \le \sum_{i=1}^{s} (d_i - 1) + 1, \tag{2.3}$$

where $d_1, \ldots, d_s$ are the degrees of the input polynomials, see [Laz83]. But such bounds are usually not tight, and plugging them into the complexity of F5 yields just a doubly exponential bound.

A related and practically relevant result is that the reverse lexicographic order leads to Gröbner bases of the lowest degree (again after a generic change of coordinates), while bases with respect to the lexicographic term order have the largest degrees. Therefore, computing degree reverse lexicographic Gröbner bases is usually the fastest. This has led to the following approach when computing Gröbner bases in practice. One first computes a degree reverse lexicographic basis, using one of the algorithms mentioned above. Then one applies a Gröbner walk algorithm [CKM97] to convert this basis to a basis of the desired term order, say lexicographic. When working with a zero-dimensional ideal, one may use the more efficient FGLM algorithm [FGLM93]. In practice, this approach is much faster than computing a lexicographic basis directly, and e.g. Magma does this automatically.

2.7 Index calculus

An important branch of cryptographic research is dedicated to studying the hardness of the discrete logarithm problem in different groups, since this determines the security of DLP-based cryptographic systems (see Section 2.1). This is usually done by means of developing algorithms that compute discrete logarithms and by studying their complexity and practical performance.

Let us assume that the goal is to compute a discrete logarithm $\ell = \log_P Q$ of an element $Q \in \langle P \rangle$ in some additively written group G (see Definition 2.1). Since we are only working in the cyclic subgroup, we write $G = \langle P \rangle$, and we let $N := |G| = \operatorname{ord}(P)$.

A combination of the Pollard–Rho Algorithm [Pol78] and the Pohlig–Hellman Algorithm [PH78] can solve any instance of the DLP in G in time $O(\sqrt{p})$, where p is the largest prime factor of N. In such an approach, if $N = \prod_{i=1}^{k} p_i^{e_i}$ is the prime factorization of N, the Pollard–Rho algorithm (or alternatively Shanks' Baby Step Giant Step Algorithm) is used to compute the discrete logarithm modulo p_i with complexity $O(\sqrt{p_i})$ for all i, and the Pohlig–Hellman Algorithm is then used to compute the discrete logarithm modulo $p_i^{e_i}$ for all i via a method called Hensel lifting, and finally modulo N via an explicit version of the Chinese Remainder Theorem.

It is thanks to the Pohlig–Hellman algorithm that cryptographers usually prefer to work in cyclic groups of prime order. Such groups yield the strongest instances of the DLP from a complexity theoretic point of view. For this reason, it is often assumed from the start that N is prime, and when dealing with attacks on the DLP, we shall do the same.

The methods of Pohlig–Hellman and Pollard–Rho work for any group G, regardless of its structure or the representation of its elements. Such algorithms are called *generic* attacks on the DLP. In fact, using a more precise definition of "generic algorithms" as given e.g. in [Sti06, Chapter 6.3], it has been shown that such algorithms never achieve better complexity than the Pollard–Rho Algorithm, namely $O(\sqrt{N})$, see [Nec94, Sho97]. Therefore, such generic attacks on the DLP are often also called *square root algorithms*.

However, when a concrete group is given, its properties can often be exploited in order to devise more efficient attacks. A particularly powerful such class of attacks are *index calculus algorithms* [EG02], which exploit the algebraic structure of the groups that they work in. There are index calculus algorithms that compute the DLP in multiplicative groups of finite fields (namely the number field sieve for prime fields [Adl79, Gor93, JL03] and the function field sieve for fields of small to medium characteristic [Cop84, Adl94, ADH94, Sch02, JL02, JL06, Jou13a, GGMZ13a, Jou13b, GGMZ13b, BBD+14, BGJT13]), elliptic curves over extension fields [Sem04, Gau09, Die11, Die13], Picard groups of hyperelliptic curves and more generally $C_{a,b}$ curves [ADH94, Gau00, Eng02, Die06, DT08, EG07, Eng08, EGT11, VJS14], and even general abelian varieties [Gau09].

The general outline of an index calculus algorithm to compute $\ell = \log_P Q$ in G is given in Algorithm 2.2. It is easy to see that this gives the correct result: Since γ is in the right

Algorithm 2.2 General outline of an index calculus algorithm

```
Input: $Q \in G = \langle P \rangle, N = |G|$
Output: $\ell = \log_P Q$
1: Factor base: Choose a factor base $\mathcal{F} = \{P_1, \dots, P_k\} \subseteq G$.
2: Relation collection: Construct relations of the form $\alpha_j P + \beta_j Q = \sum_{i=1}^{k} m_{ij} P_i$ for
   $j = 1, \dots, r > k$.
3: Linear algebra: Given the matrix $M = (m_{ij}) \in (\mathbb{Z}/N\mathbb{Z})^{k \times r}$, compute a non-zero
   column vector $\gamma = (\gamma_1, \dots, \gamma_r)^\intercal$ in the right kernel of $M$.
4: Individual logarithm: Output $\ell = -(\sum_{j=1}^{r} \alpha_j \gamma_j)(\sum_{j=1}^{r} \beta_j \gamma_j)^{-1}$ if $\sum \beta_j \gamma_j$ is invert-
   ible in $\mathbb{Z}/N\mathbb{Z}$, otherwise return to step 2.
```

kernel of M, we have $M\gamma = 0$, or equivalently

$$\sum_{j=1}^{r} m_{ij} \gamma_j = 0 \qquad \text{for all } i = 1, \dots, k.$$

Multiplying all relations from step 2 by γ_j, summing over j, and using the above equality gives

$$\sum_{j=1}^{r} \alpha_j \gamma_j P + \sum_{j=1}^{r} \beta_j \gamma_j Q = \sum_{j=1}^{r} \sum_{i=1}^{k} m_{ij} \gamma_j P_i = \sum_{i=1}^{k} \left(\sum_{j=1}^{r} m_{ij} \gamma_j \right) P_i = 0.$$

Therefore,

$$Q = -\left(\sum_{j=1}^{r} \alpha_j \gamma_j \right) \left(\sum_{j=1}^{r} \beta_j \gamma_j \right)^{-1} P = \ell P.$$

This algorithm assumes that there are efficient methods to determine whether an element $\alpha_j P + \beta_j Q$ decomposes into a sum of elements of the factor base (if so, it is called *smooth*) and to actually compute this decomposition. How this is done, and therefore the details of steps 1 and 2, depend on the properties of G, and when G contains complicated elements such as divisor classes, one of the major challenges in applying index calculus is finding suitable concepts of smoothness and decomposition.

The easiest case, and the case for which this algorithm was originally developed, is when working with integers. Then the natural choice for the factor base is the set of all primes below a certain *smoothness bound* b. A number is smooth if all of its prime factors are smaller than b, and its decomposition is obtained by computing the prime factorization. Similarly, when the group elements can be represented by polynomials (e.g. when $G = \mathbb{F}_{q^n}^\times$ for $n \gg 1$ as is the case for the function field sieve), the factor base can be taken to consist of all irreducible polynomials of degree less than some bound b. Checking for smoothness and obtaining the decomposition can then be achieved via polynomial factorization.

Index calculus in abelian varieties. The major innovation of Gaudry's index calculus algorithm for abelian varieties [Gau09] is that it computes relations algebraically. To this end, a relation is translated into a system of polynomial equations using the equations of the variety, and in order to find relations, one must find solutions to this system, for example via a Gröbner basis computation.

We sketch only the main ideas of Gaudry's algorithm. For further technical assumptions and details, see [Gau09]. We include a brief explanation of Gaudry's complexity analysis, which is for $q \to \infty$ for a family of abelian varieties with fixed parameters, i.e. the dimension

d, the number m defining the size of the representation (see below), and the degrees of the equations defining the variety and the group law are taken to be constants.

Let V be an abelian variety defined over a finite field $\mathbb{F}_q$ of dimension $d \geq 2$, and suppose we want to compute a discrete logarithm $\log_P Q$ in the group $G = V(\mathbb{F}_q)$, which has cardinality $N \approx q^d$.

For the algorithm to work, it is necessary that the elements of V and the group law are given in a representation that is convenient for computation. Therefore, we assume that there is an explicit embedding of an open subset of V into an affine space of dimension $d+m$, and hence that almost all elements of V can be written in the affine coordinates

$$(x_1, \ldots, x_d, y_1, \ldots, y_m).$$

These elements satisfy a set of m polynomial equations in the indeterminates $x_1, \ldots, x_d$, $y_1, \ldots, y_m$, and the group law is given via rational functions in these coordinates. We also assume that P and Q are representable with these coordinates. Whenever the algorithm encounters an element in its computations that is not representable by these coordinates, it is discarded, but this happens only very rarely.

The factor base in *step 1* is defined to consist of the $\mathbb{F}_q$-rational points of an absolutely irreducible one-dimensional subvariety of V (i.e. a curve in V), obtained by intersecting V with $d-1$ hyperplanes:

$$\mathcal{F} = \{(X_1, 0, \ldots, 0, Y_1, \ldots, Y_m) \in V \mid X_1, Y_1, \ldots, Y_m \in \mathbb{F}_q\}.$$

It has about q elements. It is important that the curve is not included in a proper abelian subvariety of V, so that the set generated by the elements of $\mathcal{F}$ is as large as possible.

In *step 2*, the algorithm searches for relations of the form

$$\alpha P + \beta Q = P_1 + \ldots + P_d$$

with $P_1, \ldots, P_d \in \mathcal{F}$. It chooses α, β randomly and computes $R = \alpha P + \beta Q$. Then it solves a system of polynomial equations, where the coordinates of $P_1, \ldots, P_d$ are the indeterminates. The system consists of $d+m$ equations describing the fact that the sum of the P_i is equal to R (one equation for each coordinate resulting from the formulas for the group law) and additionally a number of equations that ensure that the P_i are indeed in $\mathcal{F}$. This produces a system with more equations than unknowns, therefore it is (generically) of dimension 0. Now the system is solved by computing a Gröbner basis with respect to a lexicographic order and the subsequent factorization of a univariate polynomial. In most cases, the system does not have a solution over $\mathbb{F}_q$, since most points R do not decompose over the factor base. Whenever R decomposes, the relation can be recovered from an $\mathbb{F}_q$-solution of the system.

Since the parameters of the system (namely, the number of indeterminates, the number of equations, and the degrees of the equations) are assumed to be constant, the Gröbner basis computation can be performed in time polynomial in $\log q$ using Buchberger's algorithm. The same is true for the factorization of the univariate polynomial.

This procedure is repeated until more than $|\mathcal{F}|$ relations have been produced, i.e. until about q relations are found. The overall complexity of this step depends on the probability that a given point R can be decomposed over the factor base. Under the heuristic assumption that different unordered d-tuples of factor base elements have *different* sums, the sums of d factor base elements produce about $q^d/d!$ different elements of $V(\mathbb{F}_q)$. Since $|V(\mathbb{F}_q)| \approx q^d$, this means that one has to try out about $d!$ points R in order to find one actual relation. Therefore, it takes $d!q$ steps to produce q relations. Since d is taken to be constant and testing for or computing a decomposition is polynomial in $\log q$, the relation generation phase has a total complexity of $\tilde{O}(q)$.

The relation collection step produces a sparse matrix of size about $q \times q$ with at most d non-zero entries in each column. Using standard techniques for the resolution of large sparse linear systems, such as Lanczos' or Wiedemann's algorithm (see [Wie86, LO90]), the cost of *step 3* is $\tilde{O}(q^2)$.

The computation of the individual logarithm in *step 4* is easy, provided that $\sum \beta_j \gamma_j$ is invertible modulo N. This is true with large probability, especially if N is prime. If not, one must collect new relations, i.e. go back to step 2.

The computationally intensive steps are step 2, which has complexity $\tilde{O}(q)$, and step 3, which has complexity $\tilde{O}(q^2)$, provided that the factor base has size q. The *double large prime variation* [Thé03, Nag04, GTTD07, Die06] was designed to rebalance the complexity of these two steps and produce a better total complexity for the algorithm. It reduces the size of the factor base by defining a small proportion of $1/d$ of the original factor base elements to be "large primes". In the relation generation phase, all relations that involve more than two large primes are discarded. Then the large prime parts of the relations are eliminated to produce relations that involve no large primes. This is a bit more work than the straightforward approach would be, but it creates a smaller linear system of size only $q^{1-1/d} \times q^{1-1/d}$, thus transferring some of the cost of the linear algebra step to the relation search step. Using this trick, one obtains a complexity of $\tilde{O}(q^{2-2/d})$ for both the relation generation and the linear algebra step.

Using many heuristic arguments and assumptions, in particular that the objects the algorithm encounters behave like randomly chosen elements, the above considerations give the following theorem. Gaudry [Gau09] gives more technical details and arguments to support the heuristics, but the result remains heuristic in nature.

Theorem 2.32 ([Gau09, Heuristic result 3]). *Let us consider a family $(V_i)_{i\geq 1}$ of abelian varieties of dimension $d \geq 2$ given by explicit equations of the same form, where the cardinality of the field of definition $\mathbb{F}_{q_i}$ of V_i tends to infinity. Then there exists a probabilistic algorithm that can solve discrete logarithm problems in an abelian variety V over $\mathbb{F}_q$ in that family in heuristic time $\tilde{O}(q^{2-2/d})$. The constant in the $\tilde{O}$ depends on d and on the family, but not on q.*

Generic attacks on the DLP in $V(\mathbb{F}_q)$ have complexity $O(q^{d/2})$, since $|V(\mathbb{F}_q)| \approx q^d$. Gaudry's algorithm has lower complexity for $d \geq 3$. However, since there are large constants hidden in the O-notation of the complexity of Gaudry's attack, it can be made practical only for abelian varieties of small dimension that are given by equations in a small number of indeterminates and of small degree. Moreover, Gaudry's algorithm is expected to be more efficient in practice than generic methods only for very large values of q, and the precise crossover point is not known.

Since its publication, Gaudry's algorithm has been applied mostly to the Weil restriction of elliptic curves defined over extension fields. In fact, Gaudry suggests this application himself in his original article [Gau09]. A similar algorithm for elliptic curves was developed independently by Diem [Die11]. This algorithm of Gaudry and Diem was implemented by Joux and Vitse [JV12], and with several further improvements and variations, including a specialized implementation of the Gröbner basis algorithm F4 [JV11], they were able to solve an instance of an oracle-assisted static Diffie–Hellman problem in $E(\mathbb{F}_{2^{155}})$, which is related to but easier than the DLP in the same group [GJV10]. Faugère, Perret, Petit, and Renault [FPPR12], Petit and Quisquater [PQ12], and Shantz and Teske [ST13] study the polynomial systems that arise during this attack. They come to the conclusion that these systems are of a special shape and that special-purpose Gröbner basis techniques may lead to a significant speed-up. The application of the algorithm to Edwards curves was studied by Faugère, Gaudry, Huot, and Renault in [FGHR12, FGHR13].

Notice that this approach only threatens elliptic curves defined over extension fields and does not affect groups $E(\mathbb{F}_p)$ where p is a prime. The best attack on such groups is the Pollard–Rho attack, and the current record for computing a discrete logarithm in $E(\mathbb{F}_p)$, for p a 112-bit prime, is held by Bos, Kaihara, Kleinjung, Lenstra, and Montgomery [BKK+09], using a parallelized version of the Pollard–Rho algorithm.

Besides elliptic curves, Gaudry's algorithm for abelian varieties has been applied to the Weil restriction of hyperelliptic curves of small genus by Nagao [Nag10] and to algebraic tori by Granger and Vercauteren [GV05].

Chapter 3

Equations for the trace zero subgroup

We derive equations for the trace zero subgroup $T_n \subseteq E(\mathbb{F}_{q^n})$ of an elliptic curve E defined over $\mathbb{F}_q$. More precisely, the goal is to give a set of polynomial equations with coefficients in $\mathbb{F}_q$ such that the $\mathbb{F}_q$-rational solutions are exactly the $\mathbb{F}_q$-rational points of the trace zero variety V_n, thus corresponding to the points of T_n. However, in order to obtain a set of equations that is as simple as possible, we will be satisfied with equations that describe a large subset of T_n, and possibly a small set of extra points, as long as the exceptions are well understood. We put particular emphasis on the cases $n = 3, 5$, since these are most relevant in practice.

The equations presented here are used in Chapter 7, where we study the application of the index calculus attack of Gaudry for abelian varieties [Gau09] to the trace zero variety. This attack requires the availability of suitable equations for the variety, and its practicality depends crucially on finding convenient equations. In fact, since the index calculus attack attempts to solve the DLP in the group $V_n(\mathbb{F}_q)$, any set of equations that describes $V_n(\mathbb{F}_q)$ suffices. Therefore, we are more interested in *simple* equations with the "correct" $\mathbb{F}_q$-solutions than in equations that correctly describe V_n over $\overline{\mathbb{F}}_q$.

Finding convenient equations for $V_n(\mathbb{F}_q)$ is also an important step towards an efficient representation for the elements of T_n, since the usual approach to an efficient representation is to drop some coordinates of a point for compression and then recompute them with the help of a suitable set of equations for decompression. The equation derived in Section 3.3 might be used for such an approach, and the equation given in Section 3.4 is the basis of a more sophisticated efficient representation presented in Chapter 4.

Assumptions and Notation. *In this chapter, let T_n be the trace zero subgroup of an elliptic curve E defined over a finite field $\mathbb{F}_q$ with respect to the extension $\mathbb{F}_{q^n}|\mathbb{F}_q$ of prime degree n. Whenever we write explicit equations, we assume in addition that $\mathbb{F}_q$ does not have characteristic 2 or 3 and that E is defined by an affine Weierstraß equation*

$$E : y^2 = x^3 + Ax + B. \tag{3.1}$$

This is only for simplicity, and all our formulas may be adjusted to general finite fields and elliptic curves, see also Remark 3.9.

In order to write explicit equations, we also need a concrete representation of the field extension $\mathbb{F}_{q^n}|\mathbb{F}_q$, and for this purpose we fix the following. For the sake of simplicity, we assume that $n \mid q-1$. All of our arguments work, however, for any n and q, see also

Remark 3.1. If $n \mid q-1$, thanks to Kummer theory we can write the extension field as

$$\mathbb{F}_{q^n} = \mathbb{F}_q[\zeta]/(\zeta^n - \mu),$$

where μ is not an n-th power in $\mathbb{F}_q$. Where necessary, we take $1, \zeta, \ldots, \zeta^{n-1}$ as a polynomial basis of the field extension. When writing equations for the Weil restriction, we choose the coordinates

$$\begin{array}{rcl} x & = & x_0 + x_1\zeta + \ldots + x_{n-1}\zeta^{n-1} \\ y & = & y_0 + y_1\zeta + \ldots + y_{n-1}\zeta^{n-1}. \end{array} \tag{3.2}$$

Remark 3.1. If n does not divide $q-1$, we choose a normal basis $\{\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}\}$ of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$ and Weil restriction coordinates

$$\begin{array}{rcl} x & = & x_0\alpha + x_1\alpha^q + \ldots + x_{n-1}\alpha^{q^{n-1}} \\ y & = & y_0\alpha + y_1\alpha^q + \ldots + y_{n-1}\alpha^{q^{n-1}}. \end{array}$$

They always yield similar but more dense equations than the coordinates resulting from a polynomial basis (3.2).

Roadmap. This chapter is organized as follows. In Section 3.1, we explain how to write straightforward equations for the trace zero variety. This was already done for $n = 3$ by Frey, Naumann, and Diem, and they also show how to eliminate some variables from the system, which we briefly recall in Section 3.2. Afterwards, we derive a new equation for the trace zero subgroup from the Semaev polynomial in Section 3.3 (this equation agrees with the equation of Frey, Naumann, and Diem from Section 3.2 for $n = 3$), and a symmetrized version of this in Section 3.4. We close with a comparison of all equations in Section 3.5. Throughout, we put particular emphasis on the cases $n = 3, 5$ and write down the equations explicitly for these cases whenever possible.

3.1 Equations for the trace zero variety

Using the Weil restriction coordinates (3.2), it is not difficult to write a straightforward system of at least $n+1$ equations in $\mathbb{F}_q[x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}]$ for the affine part of V_n. Of these equations, n come from the Weil restriction of the elliptic curve and have degree 3, and at least one additional equation comes from the trace zero condition. These equations can be written down explicitly for fixed n, q, and μ.

3.1.1 Explicit equations for $n = 2$

The description of the trace zero points is particularly simple for $n = 2$.

Proposition 3.2. *The trace zero subgroup T_2 of $E(\mathbb{F}_{q^2})$ can be described as*

$$T_2 = \{(X, Y) \in E(\mathbb{F}_{q^2}) \mid X \in \mathbb{F}_q,\ Y \notin \mathbb{F}_q\} \cup E[2](\mathbb{F}_q).$$

Proof. We first prove that T_2 is contained in the union of sets on the right hand side of the equality. Let $P \in T_2$, $P \neq \mathcal{O}$, so $P = (X, Y) \in E(\mathbb{F}_{q^2})$. If $P \in E(\mathbb{F}_q)$, then $[2]P = \mathcal{O}$, hence $P \in E[2](\mathbb{F}_q)$. If $P \notin E(\mathbb{F}_q)$, then $(X, Y) = \ominus(X^q, Y^q)$. In particular $X = X^q$, so $X \in \mathbb{F}_q$, which also implies $Y \notin \mathbb{F}_q$.

To prove the other inclusion, observe that by definition $P \in E[2](\mathbb{F}_q)$ satisfies $[2]P = \mathcal{O}$, so $P \in T_2$. Let $P = (X, Y) \in E(\mathbb{F}_{q^2})$ with $X \in \mathbb{F}_q$, $Y \notin \mathbb{F}_q$. Since $X \in \mathbb{F}_q$, the points (X, Y) and $\varphi(X, Y) = (X, Y^q)$ are distinct points on E which lie on the same vertical line $x - X = 0$. Hence $(X, Y) \oplus \varphi(X, Y) = \mathcal{O}$ and $(X, Y) \in T_2$. □

In order to write down the equations, we let $\mathbb{F}_{q^2} = \mathbb{F}_q[\zeta]/(\zeta^2 - \mu)$ and $x = x_0 + x_1\zeta, y = y_0 + y_1\zeta$. Plugging this into the elliptic curve equation (3.1) gives two equations for the affine part of the Weil restriction of E:

$$\begin{aligned} y_0^2 + \mu y_1^2 &= x_0^3 + 3\mu x_0 x_1^2 + Ax_0 + B \\ 2y_0y_1 &= \mu x_1^3 + 3x_0^2 x_1 + Ax_1. \end{aligned}$$

Now thanks to Proposition 3.2, a point $(X, Y) = (X_0 + X_1\zeta, Y_0 + Y_1\zeta) \in E(\mathbb{F}_{q^2}) \setminus E(\mathbb{F}_q)$ is in T_2 if and only if $X \in \mathbb{F}_q$, i.e. if and only if $X_1 = 0$. By plugging $x_1 = 0$ into the above system we get

$$\begin{aligned} y_0^2 + \mu y_1^2 &= x_0^3 + Ax_0 + B \\ y_0y_1 &= 0 \\ x_1 &= 0, \end{aligned}$$

which describes the points of $V_2(\mathbb{F}_q)$ when $y_1 \neq 0$ (this comes from $Y \notin \mathbb{F}_q$, see Proposition 3.2). But $y_1 \neq 0$, together with the second equation, implies $y_0 = 0$. Therefore, the system simplifies to

$$\mu y_1^2 = x_0^3 + Ax_0 + B.$$

For solutions $(X_0, Y_1) \in \mathbb{F}_q^2$ with $Y_1 \neq 0$ of this equation, $(X_0, Y_1\zeta) \in E(\mathbb{F}_{q^2})$ are exactly the points of $T_2 \setminus E(\mathbb{F}_q)$. The other elements of T_2, namely those in $T_2 \cap E(\mathbb{F}_q) = E[2](\mathbb{F}_q)$, are $\mathcal{O}$ and at most three 2-torsion points $(X_0, 0)$, which are the solutions of the above equation when $Y_1 = 0$. Therefore, the solutions $(X_0, Y_1) \in \mathbb{F}_q^2$ of this equation give precisely the affine points of T_2.

Remark 3.3. Our way of deriving the system shows only that it describes T_2, which suffices for the purpose of this work. However, an argumentation similar to the one given for $n = 3$ in [Nau99, Chapter 4.2] and [Die01, Chapter 2.4.1] shows that these equations actually describe the affine part of the entire trace zero variety. See also Remark 3.4.

3.1.2 Explicit equations for $n = 3$

For $n = 3$, letting $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - \mu)$ and using (3.2), we compute the system

$$\begin{aligned} y_0^2 + 2\mu y_1 y_2 &= x_0^3 + \mu x_1^3 + \mu^2 x_2^3 + 6\mu x_0 x_1 x_2 + Ax_0 + B \\ 2y_0y_1 + \mu y_2^2 &= 3x_0^2 x_1 + 3\mu x_0 x_2^2 + 3\mu x_1^2 x_2 + Ax_1 \\ 2y_0y_2 + y_1^2 &= 3x_0^2 x_2 + 3x_0 x_1^2 + 3\mu x_1 x_2^2 + Ax_2 \\ x_1 y_2 &= x_2 y_1. \end{aligned} \tag{3.3}$$

It describes the open affine part of the trace zero variety given by the equation $x_1x_2 \neq 0$. The first 3 equations are the Weil restriction of the elliptic curve equation (3.1), as explained in Example 2.14.

The fourth equation comes from the trace zero condition: Let $(X, Y) \in E(\mathbb{F}_{q^3}) \setminus E(\mathbb{F}_q)$, which, with $X = X_0 + X_1\zeta + X_2\zeta^2$ and $Y = Y_0 + Y_1\zeta + Y_2\zeta^2$, is equivalent to X_1, X_2, Y_1, Y_2 not all $= 0$ (notice that this is true if $X_1X_2 \neq 0$). Then the points $(X, Y), (X^q, Y^q), (X^{q^2}, Y^{q^2})$ are distinct. Now $(X, Y) \in T_3$ if and only if $(X, Y) \oplus (X^q, Y^q) \oplus (X^{q^2}, Y^{q^2}) = \mathcal{O}$. It follows from the chord and tangent rule that three distinct points on an elliptic curve sum to zero if and only if they lie on a line. In our case, this means that there are a_1, a_2, a_3 not all zero such that

$$\begin{aligned} a_1X + a_2Y + a_3 &= 0 \\ a_1X^q + a_2Y^q + a_3 &= 0 \\ a_1X^{q^2} + a_2Y^{q^2} + a_3 &= 0, \end{aligned}$$

which is equivalent to

$$\det\begin{pmatrix} X & Y & 1 \\ X^q & Y^q & 1 \\ X^{q^2} & Y^{q^2} & 1 \end{pmatrix} = 0.$$

Therefore, the points of $T_3 \setminus E(\mathbb{F}_q)$ are exactly the $\mathbb{F}_{q^3}$-solutions of

$$\begin{aligned} y^2 &= x^3 + Ax + B \\ 0 &= \det\begin{pmatrix} x & y & 1 \\ x^q & y^q & 1 \\ x^{q^2} & y^{q^2} & 1 \end{pmatrix} = xy^q - x^q y - xy^{q^2} + x^{q^2}y + x^q y^{q^2} - x^{q^2}y^q \end{aligned} \tag{3.4}$$

where not both coordinates are in $\mathbb{F}_q$. Weil restriction of these two equations now yields system (3.3). Its $\mathbb{F}_q$-solutions $(X_0, X_1, X_2, Y_0, Y_1, Y_2)$ such that X_1, X_2, Y_1, Y_2 are not all $= 0$ give all points of $T_n \setminus E(\mathbb{F}_q)$. The remaining points, namely those in $T_3 \cap E(\mathbb{F}_q) = E(\mathbb{F}_q)[3]$, are the $\mathbb{F}_q$-rational 3-torsion points and can be determined easily.

For the purpose of Weil restricting equation (3.4), we observe that it can be written as

$$t + t^q + t^{q^2} = 0 \quad \text{for} \quad t := \det\begin{pmatrix} x & y \\ x^q & y^q \end{pmatrix} = xy^q - x^q y. \tag{3.5}$$

From $x = x_0 + x_1\zeta + x_2\zeta^2$, we get

$$\begin{aligned} x &= x_0 + x_1\zeta + x_2\zeta^2 \\ x^q &= x_0 + \mu^b x_1\zeta + \mu^{2b}x_2\zeta^2 \\ x^{q^2} &= x_0 + \mu^{2b}x_1\zeta + \mu^b x_2\zeta^2, \end{aligned} \tag{3.6}$$

where $b = \frac{q-1}{3}$. The second and third equalities follow from observing that we can substitute x_i for x_i^q when looking for $\mathbb{F}_q$-solutions. Analogous equations hold for y and t. Plugging this into equation (3.5) gives

$$3t_0 = 0$$

where

$$t_0 = (\mu^{2b+1} - \mu^{b+1})(x_1y_2 - x_2y_1).$$

This gives the last equation of system (3.3), since $(\mu^{2b+1} - \mu^{b+1}) \neq 0$ because μ is not a third power in $\mathbb{F}_q$.

Remark 3.4. Frey [Fre99], Naumann [Nau99], and Diem [Die01] also derive System (3.3). Again, while our way of deriving the system shows only that its $\mathbb{F}_q$-solutions are the points of $V_3(\mathbb{F}_q)$, more general arguments in [Nau99, Chapter 4.2] and [Die01, Chapter 2.4.1] show that the system has not only the correct $\mathbb{F}_q$-rational solutions but describes in fact the affine open part of the trace zero variety over $\overline{\mathbb{F}}_q$ where $x_1x_2 \neq 0$.

3.1.3 Explicit equations for $n = 5$

An analogous approach is possible for $n = 5$. We compute a system of 6 equations in 10 indeterminates $x_0, \ldots, x_4, y_0, \ldots, y_4$. The first five equations come from the elliptic curve equation and have degree 3. The last equation comes from the trace zero condition. It has degree 4 in x_1, x_2, x_3, x_4, degree 2 in y_1, y_2, y_3, y_4, and total degree 6. It is too long to be printed here, but we have computed it using Maple [Map] as follows. Examples for some fixed values of q, μ, A, B are available at `http://maikemassierer.wordpress.com/phdthesis`.

Lemma 3.5. *Let $P_1, \ldots, P_6$ be distinct points on E. Then $P_1 \oplus \ldots \oplus P_6 = \mathcal{O}$ if and only if there exists a non-trivial quadric F with $F(P_1) = \ldots = F(P_6) = 0$.*

Proof. The result follows directly from the fact that $P_1 \oplus \ldots \oplus P_6 = \mathcal{O}$ on E if and only if $P_1 + \ldots + P_6 - 6\mathcal{O}$ is the principal divisor of a function F on E (see [Was08, Theorem 11.2]), the fact that a function which has its only pole at $\mathcal{O}$ is a polynomial, and Bézout's Theorem (see [Ful08, Chapter 5.3]). □

Now let $P = (X, Y) \in E(\mathbb{F}_{q^5}) \setminus E(\mathbb{F}_q)$. As before, we derive an equation only for such trace zero points, since the points of $T_5 \cap E(\mathbb{F}_q) = E(\mathbb{F}_q)[5]$ can be determined easily. The points $P, \varphi(P), \ldots, \varphi^4(P), \mathcal{O}$ are all distinct, and $P \in T_5$ if and only if

$$P \oplus \varphi(P) \oplus \ldots \oplus \varphi^4(P) \oplus \mathcal{O} = \mathcal{O}.$$

According to Lemma 3.5, this is equivalent to there existing a non-zero quadric

$$F(x, y) = a_1x^2 + a_2xy + a_3y^2 + a_4x + a_5y + a_6$$

which passes through the points $P, \varphi(P), \ldots, \varphi^4(P), \mathcal{O}$, and the condition that $\mathcal{O}$ lies on F implies $a_3 = 0$. Such a quadric exists if and only if the system

$$\begin{array}{lclclclclcl}
a_1X^2 & + & a_2XY & + & a_4X & + & a_5Y & + & a_6 & = & 0 \\
a_1X^{2q} & + & a_2X^qY^q & + & a_4X^q & + & a_5Y^q & + & a_6 & = & 0 \\
a_1X^{2q^2} & + & a_2X^{q^2}Y^{q^2} & + & a_4X^{q^2} & + & a_5Y^{q^2} & + & a_6 & = & 0 \\
a_1X^{2q^3} & + & a_2X^{q^3}Y^{q^3} & + & a_4X^{q^3} & + & a_5Y^{q^3} & + & a_6 & = & 0 \\
a_1X^{2q^4} & + & a_2X^{q^4}Y^{q^4} & + & a_4X^{q^4} & + & a_5Y^{q^4} & + & a_6 & = & 0
\end{array}$$

has a non-trivial solution, i.e. if and only if

$$\det \begin{pmatrix} X^2 & XY & X & Y & 1 \\ X^{2q} & X^qY^q & X^q & Y^q & 1 \\ X^{2q^2} & X^{q^2}Y^{q^2} & X^{q^2} & Y^{q^2} & 1 \\ X^{2q^3} & X^{q^3}Y^{q^3} & X^{q^3} & Y^{q^3} & 1 \\ X^{2q^4} & X^{q^4}Y^{q^4} & X^{q^4} & Y^{q^4} & 1 \end{pmatrix} = 0.$$

Hence the equation describing the trace zero condition is

$$\det \begin{pmatrix} x^2 & xy & x & y & 1 \\ x^{2q} & x^qy^q & x^q & y^q & 1 \\ x^{2q^2} & x^{q^2}y^{q^2} & x^{q^2} & y^{q^2} & 1 \\ x^{2q^3} & x^{q^3}y^{q^3} & x^{q^3} & y^{q^3} & 1 \\ x^{2q^4} & x^{q^4}y^{q^4} & x^{q^4} & y^{q^4} & 1 \end{pmatrix} = t + t^q + t^{q^2} + t^{q^3} + t^{q^4} = 0,$$

where

$$t = \det \begin{pmatrix} x^2 & xy & x & y \\ x^{2q} & x^qy^q & x^q & y^q \\ x^{2q^2} & x^{q^2}y^{q^2} & x^{q^2} & y^{q^2} \\ x^{2q^3} & x^{q^3}y^{q^3} & x^{q^3} & y^{q^3} \end{pmatrix}.$$

Weil restriction yields $5t_0 = 0$, where t_0 is a polynomial of degree 6 in $x_1, \ldots, x_4, y_1, \ldots, y_4$.

3.2 Frey's equations for $n = 3$

Frey [Fre99, Section 3.2], Naumann [Nau99, Chapter 4.2], and Diem [Die01, Chapter 2.4.1] study the equations from Section 3.1.2 in more detail. They show how the system can be manipulated in order to eliminate the indeterminates y_1 and y_2. Hence, an open affine part of the trace zero variety can be described by just two equations:

$$\begin{aligned}(3x_0^2 + 3\mu x_1 x_2 + A)^2 &= 12x_0(x_0^3 + \mu x_1^3 + \mu^2 x_2^3 + Ax_0 + B) \\ y_0^2 &= x_0^3 + \mu x_1^3 + \mu^2 x_2^3 + Ax_0 + B.\end{aligned} \tag{3.7}$$

Remark 3.6. Naumann suggests to use equation (3.7) for an efficient representation of the points in T_3, see Chapter 4.1.

In principle, such an approach works also for larger n. However, the clever manipulation of such a system with $2n$ indeterminates becomes very difficult already for $n = 5$. The elimination of indeterminates with the help of a computer (e.g. using Gröbner bases) is also very costly. Therefore in the following section, we propose a method that directly produces an equation only in the x-coordinates. We will see that for $n = 3$, our method produces the same equation as Frey, Naumann, and Diem, namely, equation (3.7) is the same as equation (3.9).

3.3 The Semaev equation

In this section we use Semaev's summation polynomials [Sem04] to write an equation for the set of $\mathbb{F}_q$-rational points of the trace zero variety in the x-coordinates only. Proposition 3.2 shows that the case $n = 2$ is particularly simple. Therefore *in this and the following section, we concentrate only on the case $n \geq 3$.*

Semaev introduced *summation polynomials* in the context of attacking the elliptic curve discrete logarithm problem. They give polynomial conditions describing when a fixed number of points on an elliptic curve sum to $\mathcal{O}$, involving only the x-coordinates of the points.

Definition 3.7. Let $\mathbb{F}_q$ be a finite field of characteristic different from 2 and 3, and let E be a smooth elliptic curve defined by the affine equation (3.1) with coefficients $A, B \in \mathbb{F}_q$. Define the m-th *summation polynomial* f_m recursively by

$$\begin{aligned} f_2(z_1, z_2) &= z_1 - z_2 \\ f_3(z_1, z_2, z_3) &= (z_1 - z_2)^2 z_3^2 - 2((z_1 + z_2)(z_1 z_2 + A) + 2B)z_3 \\ &\quad +(z_1 z_2 - A)^2 - 4B(z_1 + z_2) \\ f_m(z_1, \ldots, z_m) &= \mathrm{Res}_z(f_{m-k}(z_1, \ldots, z_{m-k-1}, z), f_{k+2}(z_{m-k}, \ldots, z_m, z)) \end{aligned}$$

for $m \geq 4$ and $m - 3 \geq k \geq 1$, where Res denotes the resultant.

We briefly recall the properties of summation polynomials that we will need.

Theorem 3.8 ([Sem04, Theorem 1])**.** *For any $m \geq 2$, let $Z_1, \ldots, Z_m$ be elements of the algebraic closure $\overline{\mathbb{F}}_q$ of $\mathbb{F}_q$. Then $f_m(Z_1, \ldots, Z_m) = 0$ if and only if there exist $Y_1, \ldots, Y_m \in \overline{\mathbb{F}}_q$ such that the points (Z_i, Y_i) are on E and*

$$(Z_1, Y_1) \oplus \ldots \oplus (Z_m, Y_m) = \mathcal{O}$$

in the group $E(\overline{\mathbb{F}}_q)$. Furthermore, f_m has degree 2^{m-2} in each variable and total degree $(m-1)2^{m-2}$, and it is absolutely irreducible. For $m \geq 3$, the polynomial f_m is symmetric.

Remark 3.9. Definition 3.7 is the original definition that Semaev gave in [Sem04]. Semaev polynomials can be defined and computed also over a finite field of characteristic 2 or 3. Although the formulas look different, the properties are analogous to those stated in Theorem 3.8. Hence all results of this and the following section hold, with the appropriate adjustments, over a finite field of any characteristic.

Since the points in T_n are characterized by the condition that their Frobenius conjugates sum to zero, we can use the Semaev polynomial to give an equation only in x. It is clear that $(X,Y)\in T_n$ implies $f_n(X,X^q,\ldots,X^{q^{n-1}})=0$. The opposite implication has some obvious exceptions.

Lemma 3.10. *For any prime n, let T_n denote the trace zero subgroup associated with the field extension $\mathbb{F}_{q^n}|\mathbb{F}_q$. We have*

$$\bigcup_{k=0}^{\lfloor\frac{n}{2}\rfloor-1}(E[n-2k](\mathbb{F}_q)\oplus E[2]\cap T_n)\subseteq\{(X,Y)\in E(\mathbb{F}_{q^n})\mid f_n(X,X^q,\ldots,X^{q^{n-1}})=0\}\cup\{\mathcal{O}\}.$$

Proof. Let $k\in\{0,\ldots,\lfloor\frac{n}{2}\rfloor\}$, and let $P=Q\oplus R$ with $Q\in E[n-2k](\mathbb{F}_q), R\in E[2]\cap T_n$. Then we have

$$\begin{aligned}
&\underbrace{P\oplus\varphi(P)\oplus\ldots\oplus\varphi^{n-2k-1}(P)}_{n-2k\text{ summands}}\oplus\underbrace{\varphi^{n-2k}(P)\ominus\varphi^{n-2k+1}(P)\oplus\ldots\ominus\varphi^{n-1}(P)}_{2k\text{ summands with alternating signs}}\\
=\;&\underbrace{Q\oplus\ldots\oplus Q}_{n-2k\text{ summands}}\oplus\underbrace{Q\ominus Q\oplus\ldots\ominus Q}_{2k\text{ summands with alternating signs}}\oplus R\oplus\varphi(R)\oplus\ldots\oplus\varphi^{n-1}(R)\\
=\;&[n-2k]Q\oplus\mathrm{Tr}(R)\\
=\;&\mathcal{O},
\end{aligned}$$

where for the first equality, we have used that $Q\in E(\mathbb{F}_q)$ and $R\in E[2]$, and for the third equality, we have used that $Q\in E[n-2k]$ and $R\in T_n$. □

Notice that the points of the form $P=Q\oplus R$ with $Q\in E[n-2k](\mathbb{F}_q)$ and $R\in E[2]\cap T_n$ are not trace zero points if $Q\neq\mathcal{O}$ and $3\leq n-2k\leq n-2$. For the interesting cases $n=3$ and 5 we prove that these are the only exceptions.

Proposition 3.11. *Let T_n be the trace zero subgroup associated with the field extension $\mathbb{F}_{q^n}|\mathbb{F}_q$. We have*

$$\begin{aligned}
T_3 &= \{(X,Y)\in E(\mathbb{F}_{q^3})\mid f_3(X,X^q,X^{q^2})=0\}\cup\{\mathcal{O}\}\\
T_5\cup(E[3](\mathbb{F}_q)\oplus(E[2]\cap T_5)) &= \{(X,Y)\in E(\mathbb{F}_{q^5})\mid f_5(X,X^q,\ldots,X^{q^4})=0\}\cup\{\mathcal{O}\}.
\end{aligned}$$

Proof. Let $P=(X,Y)\in E(\mathbb{F}_{q^3})$ with $f_3(X,X^q,X^{q^2})=0$. Then by the properties of the Semaev polynomial, there exist $Y_0,Y_1,Y_2\in\overline{\mathbb{F}}_q$ such that $(X,Y_0)\oplus(X^q,Y_1)\oplus(X^{q^2},Y_2)=\mathcal{O}$. Obviously we have $Y_i=Y^{q^i}$ or $Y_i=\ominus Y^{q^i}, i=0,1,2$, so $P\pm\varphi(P)\pm\varphi^2(P)=\mathcal{O}$. We have to show that all signs are "$\oplus$". Suppose $P\ominus\varphi(P)\oplus\varphi^2(P)=\mathcal{O}$. By applying φ, we get $\varphi(P)\ominus\varphi^2(P)\oplus P=\mathcal{O}$. Adding these two equations gives $[2]P=\mathcal{O}$, implying that $P=\ominus P$, hence $P\oplus\varphi(P)\oplus\varphi^2(P)=\mathcal{O}$. In particular, $P\in T_3$. The rest follows by symmetry.

Now let $P=(X,Y)\in E(\mathbb{F}_{q^5})$ with $f_5(X,X^q,\ldots,X^{q^4})=0$. Then as before, $P\pm\varphi(P)\pm\varphi^2(P)\pm\varphi^3(P)\pm\varphi^4(P)=\mathcal{O}$. If all signs are "$\oplus$", then $P\in T_5$. We treat all other cases below.

[one minus] Assume $P \oplus \varphi(P) \oplus \varphi^2(P) \oplus \varphi^3(P) \ominus \varphi^4(P) = \mathcal{O}$. Applying φ to the equation and adding the two equations, we get $[2]\varphi(P) \oplus [2]\varphi^2(P) \oplus [2]\varphi^3(P) = \mathcal{O}$, and by substituting into twice the first equation, $[2]P = \varphi^4([2]P)$. Hence $[2]P \in E(\mathbb{F}_{q^4}) \cap E(\mathbb{F}_{q^5}) = E(\mathbb{F}_q)$, so $[2]P \in E[3](\mathbb{F}_q)$. Now $P = Q \oplus R \in E[6]$ is the sum of $Q \in E[3]$ and $R \in E[2]$. We have $Q = \ominus 2Q = \ominus 2P \in E[3](\mathbb{F}_q)$. From the original equation $P \oplus \varphi(P) \oplus \varphi^2(P) \oplus \varphi^3(P) \ominus \varphi^4(P) = \mathcal{O}$, we get an analogous equation in R, which together with $R \in E[2]$ gives $R \in T_5$.

[two minuses in a row] Assume $P \oplus \varphi(P) \oplus \varphi^2(P) \ominus \varphi^3(P) \ominus \varphi^4(P) = \mathcal{O}$. Applying φ^2 and adding, we get $[2]\varphi^2(P) = \mathcal{O}$, hence $P = \ominus P$ and therefore $P \in T_5$.

[two minuses not in a row] Finally, assume $P \oplus \varphi(P) \ominus \varphi^2(P) \oplus \varphi^3(P) \ominus \varphi^4(P) = \mathcal{O}$. Applying φ and adding, we get $[2]\varphi(P) = \mathcal{O}$, hence $P = \ominus P$ and therefore $P \in T_5$.

The other cases follow by symmetry. $\square$

Remark 3.12. The polynomial f_n is a convenient equation for T_n, since in practice, for any root $X \in \mathbb{F}_{q^n}$ of $f_n(x, x^q, \ldots, x^{q^{n-1}})$ we can decide efficiently whether $(X, Y) \in T_n$.

For $n = 3$ we only need to check that $Y \in \mathbb{F}_{q^3}$. This guarantees that $(X, Y) \in T_3$, by Proposition 3.11.

For $n = 5$, by Proposition 3.11 we have to exclude from the solutions of $f_5 = 0$ the points $(X, Y) \in E$ such that $Y \notin \mathbb{F}_{q^5}$ and the points of the form $Q \oplus R$ where $\mathcal{O} \neq Q \in E[3](\mathbb{F}_q)$ and $R \in E[2] \cap T_5$. Let $\mathcal{L}$ be the set of the x-coordinates of the elements $Q \oplus R \in E[3](\mathbb{F}_q) \oplus (E[2] \cap T_5)$ with $Q \neq \mathcal{O}$. Then $\mathcal{L}$ has cardinality at most 16. A root $X \in \mathbb{F}_{q^5}$ of $f_5(x, x^q, \ldots, x^{q^4})$ corresponds to a point $(X, Y) \in T_5$ if and only if $X \notin \mathcal{L}$ and $Y \in \mathbb{F}_{q^5}$.

Lemma 3.13. *Let $h \in \mathbb{F}_q[x_0, \ldots, x_{n-1}]$ be a polynomial with $h(X_0, \ldots, X_{n-1}) = 0$ for all $(X_0, \ldots, X_{n-1}) \in \mathbb{F}_q^n$, and assume that $\deg_{x_i}(h) < q$ for $i \in \{0, \ldots, n-1\}$. Then h is the zero polynomial.*

Proof. Write

$$V(h) = \{(X_0, \ldots, X_{n-1}) \in \overline{\mathbb{F}}_q^n \mid h(X_0, \ldots, X_{n-1}) = 0\} \subseteq \overline{\mathbb{F}}_q^n$$

for the zero locus of h over the algebraic closure of $\mathbb{F}_q$ and

$$I(V) = \{f \in \mathbb{F}_q[x_0, \ldots, x_{n-1}] \mid f(X_0, \ldots, X_{n-1}) = 0 \text{ for all } (X_0, \ldots, X_{n-1}) \in V\}$$

for the ideal of the polynomials vanishing on some $V \subseteq \overline{\mathbb{F}}_q^n$.

First we show that $I(\mathbb{F}_q^n) = J_n$ where $J_n = (x_0^q - x_0, \ldots, x_{n-1}^q - x_{n-1})$. We proceed by induction on n. The claim holds for $n = 1$ since the elements of $\mathbb{F}_q$ are exactly those elements of $\overline{\mathbb{F}}_q$ that satisfy the equation $x_0^q - x_0$. Assuming that the statement is true for $n - 1$, we have

$$\begin{aligned}
I(\mathbb{F}_q^n) &= \bigcap_{(\alpha_0, \ldots, \alpha_{n-1}) \in \mathbb{F}_q^n} (x_0 - \alpha_0, \ldots, x_{n-1} - \alpha_{n-1}) \\
&= \bigcap_{\alpha_0 \in \mathbb{F}_q} \bigcap_{(\alpha_1, \ldots, \alpha_{n-1}) \in \mathbb{F}_q^{n-1}} (x_0 - \alpha_0, \ldots, x_{n-1} - \alpha_{n-1}) \\
&= \bigcap_{\alpha_0 \in \mathbb{F}_q} (x_0 - \alpha_0, x_1^q - x_1, \ldots, x_{n-1}^q - x_{n-1}) \\
&= \left(\prod_{\alpha_0 \in \mathbb{F}_q} (x_0 - \alpha_0), x_1^q - x_1, \ldots, x_{n-1}^q - x_{n-1} \right) \\
&= J_n.
\end{aligned}$$

Now we show that $h = 0$. Since h vanishes on $\mathbb{F}_q^n$, we have $\mathbb{F}_q^n \subseteq V(h) \subseteq \overline{\mathbb{F}}_q^n$, which implies $h \in I(V(h)) \subseteq I(\mathbb{F}_q^n) = J_n$. The leading terms of $x_0^q - x_0, \dots, x_{n-1}^q - x_{n-1}$ with respect to any term order are $x_0^q, \dots, x_{n-1}^q$, in particular they are pairwise coprime. Hence the polynomials $x_0^q - x_0, \dots, x_{n-1}^q - x_{n-1}$ are a Gröbner basis of J_n. Therefore, $h \in J_n$ implies that h reduces to zero using the generators of J_n, i.e. if we divide h by $x_i^q - x_i$ whenever the leading term of h is divisible by x_i^q, we must obtain remainder zero when no more division is possible. But since $\deg_{x_i}(h) < q$ for all i, h is equal to the remainder of the division of h by $x_0^q - x_0, \dots, x_{n-1}^q - x_{n-1}$, hence $h = 0$. □

The x-coordinates of the points of $V_n(\mathbb{F}_q)$ correspond to zeros of the Weil restriction of $f_n(x, \dots, x^{q^{n-1}})$. Notice that since we are only interested in $\mathbb{F}_q$-solutions of the equations resulting from the Weil restriction, we may reduce all equations modulo $x_i^q - x_i$ for $i = 0, \dots, n-1$. Then all resulting equations have degree less than q in each indeterminate. Now Lemma 3.13 implies that the last $n-1$ equations disappear, since the trace of any $\mathbb{F}_{q^n}$-rational point is $\mathbb{F}_q$-rational. Therefore, although Weil restriction could produce up to n equations, by reducing modulo the equations $x_i^q - x_i$ we obtain only one equation at the end. We denote this new equation by

$$\tilde{f}_n(x_0, \dots, x_{n-1}) = 0. \tag{3.8}$$

We stress that its $\mathbb{F}_q$-solutions are x-coordinates of $\mathbb{F}_q$-points of the trace zero variety, together with some extra points described in Lemma 3.10 and Proposition 3.11. In Remark 3.12 we discussed how to distinguish the extra solutions. Conveniently, the q-th powers disappear thanks to the reduction modulo $x_i^q - x_i$, and we are left with an equation $\tilde{f}_n$ of the same degree as the original Semaev polynomial f_n.

Summarizing, compared to the straightforward system of equations describing the trace zero variety (as explained in Section 3.2, these are at least $n+1$ equations in the $2n$ indeterminates x_i and y_i), we are able to eliminate all y-coordinates and all equations but one, leaving us with only one equation in n indeterminates. We pay for this with a slightly higher degree, see Section 3.5 for a detailed comparison.

3.3.1 Explicit equations for $n = 3$

For $n = 3$ and $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - \mu)$, we have

$$\begin{aligned} f_3(x, x^q, x^{q^2}) \quad = \quad & x^{2q^2+2q} - 2x^{2q^2+q+1} + x^{2q^2+q} - 2x^{q^2+2q+1} - 2x^{q^2+q+2} - 2Ax^{q^2+q} \\ & -2Ax^{q^2+1} - 4Bx^{q^2} + x^{2q+2} - 2Ax^{q+1} - 4Bx^q - 4Bx + A^2. \end{aligned}$$

For Weil restriction, we use (3.6), which gives

$$\begin{aligned} \tilde{f}_3(x_0, x_1, x_2) \quad = \quad & -3x_0^4 - 12\mu^2 x_0 x_2^3 - 12\mu x_0 x_1^3 + 18\mu x_0^2 x_1 x_2 \\ & +9\mu^2 x_1^2 x_2^2 - 6Ax_0^2 + 6A\mu x_1 x_2 - 12Bx_0 + A^2. \end{aligned} \tag{3.9}$$

As explained above, this equation is equal to the one computed by Frey, Naumann, and Diem, i.e. equation (3.7). The advantage of our approach is, however, that we are able to produce this equation only in the x-coordinates in a direct way, without first computing and eliminating from a system of equations for the trace zero variety and that it works also for larger n.

3.3.2 Explicit equations for $n = 5$

The fifth Semaev polynomial f_5 is too big to be printed here, but a computer program can easily work with it, in fact we did this in Magma [BCP97]. It has total degree 32 and degree 8 in each indeterminate. Computing $\tilde{f}_5(x_0, \ldots, x_4)$ takes some time (about 7 hours with Magma). It has total degree 32, degree 32 in x_0 and degree 31 in $x_1, \ldots, x_4$. Examples are available at `http://maikemassierer.wordpress.com/phdthesis` for some fixed values of q, μ, A, B.

3.4 The symmetrized Semaev equation

We take the approach from the previous section a step further in order to produce an even more convenient equation for the x-coordinates of trace zero points. As before, *we consider only the case* $n \geq 3$.

As the Semaev polynomials are symmetric in nature, they can be written in terms of the symmetric functions. This fact was first exploited by Gaudry in the context of index calculus for elliptic curves [Gau04]. We write

$$f_n(z_1, \ldots, z_n) = g_n(e_1(z_1, \ldots, z_n), \ldots, e_n(z_1, \ldots, z_n)), \tag{3.10}$$

where e_i are the elementary symmetric polynomials

$$e_i(z_1, \ldots, z_n) = \sum_{1 \leq j_1 < \ldots < j_i \leq n} z_{j_1} \cdot \ldots \cdot z_{j_i},$$

and call g_n the *"symmetrized" n-th Semaev polynomial.* The advantage over the original Semaev polynomial is that g_n has lower degree (it has total degree 2^{n-2}, while f_n has total degree $(n-1)2^{n-2}$) and fewer $\mathbb{F}_{q^n}$-solutions, as it respects the inherent symmetry of the sum (i.e. where f_n has as solutions all permutations of possible x-coordinates, g_n has only one solution, the symmetric functions of these coordinates). See [JV12] for how to efficiently compute the symmetrized Semaev polynomials. In this sense,

$$g_n(s_1, \ldots, s_n) = 0 \tag{3.11}$$

also describes the points of T_n via the relations

$$s_i = e_i(x, x^q, \ldots, x^{q^{n-1}}), \quad i = 1, \ldots, n.$$

Notice that for $X \in \mathbb{F}_{q^n}$, we have $e_i(X, X^q, \ldots, X^{q^{n-1}}) \in \mathbb{F}_q$. Summarizing, g_n is a polynomial with $\mathbb{F}_q$-coefficients by equation (3.10), as well as the polynomials $\tilde{e}_i$ that we obtain by Weil descent from the symmetric functions in the q-powers of x:

$$s_i = \tilde{e}_i(x_0, \ldots, x_{n-1}), \; i = 1, \ldots, n. \tag{3.12}$$

Furthermore, we get exactly one new relation per equation (reducing modulo $x_i^q - x_i$ and applying Lemma 3.13, as before). Hence we have a total of n equations in the coordinates of the Weil restriction describing the symmetric functions. The q-th powers in the exponents disappear thanks to the reduction, and each $\tilde{e}_i$ is homogeneous of degree i.

3.4.1 Explicit equations for $n = 3$

The symmetrized third Semaev polynomial is

$$g_3(s_1, s_2, s_3) = s_2^2 - 4s_1s_3 - 4Bs_1 - 2As_2 + A^2 \tag{3.13}$$

and describes the trace zero subgroup via

$$\begin{array}{rclcl} s_1 &=& x + x^q + x^{q^2} &=& 3x_0 \\ s_2 &=& x^{1+q} + x^{1+q^2} + x^{q+q^2} &=& 3x_0^2 - 3\mu x_1x_2 \\ s_3 &=& x^{1+q+q^2} &=& x_0^3 - 3\mu x_0x_1x_2 + \mu x_1^3 + \mu^2 x_2^3. \end{array} \tag{3.14}$$

3.4.2 Explicit equations for $n = 5$

The symmetrized fifth Semaev polynomial has total degree 8. It has degree 6 in the first, third, and fifth indeterminate and degree 8 in the second and fourth indeterminate. We can compute it efficiently with Magma (about 10 seconds with a Gröbner basis elimination). It has a small number of terms compared to the original polynomial, but printing it here would still take several pages. Some examples for fixed values of q, μ, A, B can be downloaded from http://maikemassierer.wordpress.com/phdthesis. The Weil restriction of the symmetric functions is

$$\begin{array}{rcl} s_1 &=& 5x_0 \\ s_2 &=& 10x_0^2 - 5\mu x_1x_4 - 5\mu x_2x_3 \\ s_3 &=& 10x_0^3 + 5\mu^2x_3^2x_4 + 5\mu^2x_2x_4^2 + 5\mu x_1x_2^2 + 5\mu x_1^2x_3 - 15\mu x_0x_1x_4 - 15\mu x_0x_2x_3 \\ s_4 &=& 5x_0^4 - 15\mu x_0^2x_1x_4 - 15\mu x_0^2x_2x_3 - 5\mu x_1^3x_2 - 5\mu^2x_1x_3^3 - 5\mu^2x_2^3x_4 - 5\mu^3x_3x_4^3 \\ && +5\mu^2x_2^2x_3^2 + 5\mu^2x_1^2x_4^2 + 10\mu x_0x_1^2x_3 + 10\mu x_0x_1x_2^2 + 10\mu^2x_0x_3^2x_4 + 10\mu^2x_0x_2x_4^2 \\ && -5\mu^2x_1x_2x_3x_4 \\ s_5 &=& x_0^5 + \mu^3x_3^5 + \mu^4x_4^5 + \mu x_1^5 + \mu^2x_2^5 - 5\mu^2x_1x_2^3x_3 - 5\mu^3x_1x_2x_4^3 - 5\mu^3x_2x_3^3x_4 \\ && -5\mu x_0x_1^3x_2 - 5\mu^2x_0x_1x_3^3 - 5\mu^2x_0x_2^3x_4 - 5\mu^3x_0x_3x_4^3 - 5\mu^2x_1^3x_3x_4 - 5\mu x_0^3x_1x_4 \\ && -5\mu x_0^3x_2x_3 + 5\mu x_0^2x_1^2x_3 + 5\mu x_0^2x_1x_2^2 + 5\mu^2x_0^2x_2x_4^2 + 5\mu^2x_0^2x_3^2x_4 + 5\mu^2x_0x_1^2x_4^2 \\ && +5\mu^2x_0x_2^2x_3^2 + 5\mu^2x_1^2x_2^2x_4 + 5\mu^2x_1^2x_2x_3^2 + 5\mu^3x_1x_3^2x_4^2 + 5\mu^3x_2^2x_3x_4^2 \\ && -5\mu^2x_0x_1x_2x_3x_4. \end{array}$$

3.5 Comparison

The equations given in this chapter describe different sets: The equations from Sections 3.1 and 3.2 describe open subsets of the trace zero variety, whereas the equations from Sections 3.3 and 3.4 describe only the x-coordinates of trace zero points. In particular, an $\mathbb{F}_q$-solution of one of the Semaev equations is not guaranteed to produce a point of T_n because the corresponding y-coordinate may be in the wrong field. Nevertheless, the simpler Semaev equations prove particularly useful for finding an efficient representation and applying index calculus to trace zero subgroups (see Chapters 4 and 7). Therefore, we give a brief comparison of all (systems of) equations below. We compare

1. the system of equations for the trace zero variety from Section 3.1,
2. the system of equations given by Frey for $n = 3$, as presented in Section 3.2,
3. the Semaev equation from Section 3.3, and
4. the symmetrized Semaev equation from Section 3.4.

Table 3.1 Comparison of equations for $n = 3$

$n = 3$	System 1	System 2	System 3	System 4
number of equations	4	2	1	1
number of indeterminates	6	4	3	3
total degree	3	4	4	2
degree in x_0	3	4	4	–
degree in x_1	3	3	3	–
degree in x_2	3	3	3	–
degree in y_0	2	2	–	–
degree in y_1	2	–	–	–
degree in y_2	2	–	–	–
degree in s_0	–	–	–	1
degree in s_1	–	–	–	2
degree in s_2	–	–	–	1

Table 3.2 Comparison of equations for $n = 5$

$n = 5$	System 1	System 3	System 4
number of equations	6	1	1
number of indeterminates	10	5	5
total degree	6	32	8
degree in x_0	3	32	–
degree in x_1	4	31	–
degree in x_2	4	31	–
degree in x_3	4	31	–
degree in x_4	4	31	–
degree in y_0	2	–	–
degree in y_1	2	–	–
degree in y_2	2	–	–
degree in y_3	2	–	–
degree in y_4	2	–	–
degree in s_0	–	–	6
degree in s_1	–	–	8
degree in s_2	–	–	6
degree in s_3	–	–	8
degree in s_4	–	–	6

We consider the cases $n = 3$ and $n = 5$ separately in Tables 3.1 and 3.2, respectively, and we are particularly interested in the degrees of the equations.

The degree, the number of indeterminates, and hence also the number of terms grows quickly with n. Therefore, the equations are difficult to compute even for relatively small n. The degree of the Semaev polynomial grows exponentially in n (precisely, it is $(n-1)2^{n-2}$), which makes the polynomial hard to compute already for $n \geq 8$. Although the degree of the symmetrized Semaev polynomial is only 2^{n-2}, it still depends exponentially on n. This is the main drawback of the Semaev equations. The major advantage, however, is that the Semaev polynomials describe the trace zero subgroup using only half as many variables as the standard equations (n compared to $2n$) and only one equation (as compared to $n+1$).

Chapter 4

Point compression over small degree extension fields

In the conclusions of [AC07], large bandwidth is mentioned as the only drawback of using trace zero subgroups in pairing-based cryptography. In this chapter we solve this problem by finding an optimal representation for the elements of the trace zero subgroup of an elliptic curve. It builds on the symmetrized Semaev equation for the trace zero subgroup presented in Chapter 3.4, and in principle it works for trace zero subgroups of elliptic curves with respect to any small prime extension degree n. It can be made practical for $n = 3, 5$.

Two representations that work for these parameters have already been proposed. Naumann [Nau99, Chapter 5.2.3] suggests to use equation (3.7) for an efficient representation for $n = 3$, see Section 4.1, and Silverberg [Sil05] gives a different representation for $n = 3, 5$. The representation we propose here is new and conceptually different from the existing ones. Our compression and decompression algorithms are more efficient than those of both [Nau99] and [Sil05], and points are recovered with smaller ambiguity. In addition, our representation is (to the extent of our knowledge) the only one that is compatible with scalar multiplication of points, which is the only operation needed in many DLP-based cryptographic protocols such as Diffie–Hellman.

Assumptions and Notation. *In this chapter, let $\mathbb{F}_q$ be a finite field with q elements, and let E be an elliptic curve defined over $\mathbb{F}_q$ by an affine Weierstraß equation. We study the trace zero subgroup T_n of $E(\mathbb{F}_{q^n})$, where n is prime.*

Remark 4.1. The content of this chapter builds on the equations derived from the Semaev polynomial in Chapter 3. They are written down explicitly only for elliptic curves with short Weierstraß equation

$$E : y^2 = x^3 + Ax + B$$

over fields of characteristic not equal to 2 or 3. We often use those equations here, thus implicitly assuming that $\mathbb{F}_q$ has characteristic at least 5. However, as explained in Remark 3.9, such equations can be obtained for elliptic curves defined over a field of any characteristic. Hence all results of this chapter also hold, with the appropriate adjustments, over a finite field of any characteristic.

We recall from Chapter 2.5 that it is customary to represent a point $(X, Y) \in E(\mathbb{F}_{q^n})$ via its x-coordinate $X \in \mathbb{F}_{q^n}$ and that this yields an optimal representation of $E(\mathbb{F}_{q^n})$ (see Example 2.23). Since a point $P = (X, Y) \in T_n$ is an element of $E(\mathbb{F}_{q^n})$, we can also represent P via $X \in \mathbb{F}_{q^n}$. This representation however is not optimal, since $|T_n| \approx q^{n-1} \not\approx q^n$ for large q (see Remark 2.25). In this chapter we find a representation for the elements of T_n via $n-1$ coordinates in $\mathbb{F}_q$. This is optimal.

Recall also that the representation mentioned above identifies pairs of points, since P and $\ominus P$ have the same x-coordinate. We often say that the x-coordinate is a representation for the *equivalence class* consisting of P and $\ominus P$. The representation that we propose here identifies a small number of points as well.

Before we present our new representation, we notice that the case $n = 2$ allows a trivial optimal representation for the elements of T_n. This follows directly from Proposition 3.2. Hence in the next sections we concentrate on the more interesting case of *odd* primes n.

Corollary 4.2. *Representing a point $(X, Y) \in T_2$ by X yields a representation of optimal size.*

Proof. From Proposition 3.2, we know that

$$T_2 = \{(X, Y) \in E(\mathbb{F}_{q^2}) \mid X \in \mathbb{F}_q,\ Y \notin \mathbb{F}_q\} \cup E[2](\mathbb{F}_q).$$

This means that we have $X \in \mathbb{F}_q$ for all $(X, Y) \in T_2$. This representation is optimal, since $|T_2| \approx q$. □

Since it is our goal to find a representation for the elements of T_n via $n-1$ coordinates in $\mathbb{F}_q$, it is convenient to use the identification $T_n = V_n(\mathbb{F}_q)$ via the Weil restriction coordinates. Whenever writing explicit equations in these coordinates, we need to choose a basis of the extension $\mathbb{F}_{q^n}|\mathbb{F}_q$. For this purpose, we fix the following.

Assumptions and Notation. *We use the same notation as in Chapter 3. Specifically, we assume that $n > 2$ (we may do this thanks to Corollary 4.2) and that*

$$\mathbb{F}_{q^n} = \mathbb{F}_q[\zeta]/(\zeta^n - \mu)$$

is a Kummer extension of $\mathbb{F}_q$. We use the Weil restriction coordinates (3.2).

As explained in Chapter 3.1, these coordinates may be used to write a total of $n+1$ equations or more in $\mathbb{F}_q[x_0, \dots, x_{n-1}, y_0, \dots, y_{n-1}]$ defining the affine part of V_n. The $\mathbb{F}_q$-rational points of V_n correspond in a natural way to the points of T_n (which are $\mathbb{F}_{q^n}$-rational points of E), and the group law on E translates directly into V_n.

When $\mathbb{F}_{q^n}$ cannot be written as a Kummer extension, we choose a normal basis, see Remark 3.1. This produces equations for the affine part of V_n with the same properties.

With this notation, the coordinates $(x_0, \dots, x_{n-1}, y_0, \dots, y_{n-1})$ are the obvious way of representing points in $V_n(\mathbb{F}_q)$, but this representation is far from optimal in size. By dropping the y_i, we obtain a representation of size n, which is still not optimal, as discussed in Remark 2.25. It is our goal to find a representation using exactly $n-1$ coordinates in $\mathbb{F}_q$.

Roadmap. The rest of this chapter is organized as follows. First we briefly recall the representation for $n = 3$ from [Nau99] in Section 4.1. Then in Section 4.2, using the symmetrized Semaev equation from Chapter 3.4, we propose a new representation for the points on the trace zero variety. The size of the representation is optimal, and we give efficient compression and decompression algorithms in Section 4.3. We briefly discuss arithmetic in compressed coordinates in Section 4.4. In Section 4.5 we analyze in detail what our method produces for the cases $n = 3$ and 5. We give explicit equations and concrete examples computed with Magma. It is generally agreed that 3 and 5 are the practically relevant extension degrees in the case of elliptic curves (see e.g. [Lan04b]).

4.1 Naumann's representation

When constructing an efficient representation for the points of T_n using $n-1$ coordinates in $\mathbb{F}_q$, it is a natural idea to search for a suitable equation in the coordinates $x_0, \dots, x_{n-1}$, since such an equation is compatible with dropping the y-coordinate. Then the compression algorithm can simply drop one of the x_i, and the decompression algorithm can use the equation to recompute it. When the equation does not have degree 1 in the chosen x_i, the missing coordinate cannot be recovered uniquely. In such a case, it is necessary to use some extra bits ν to remember which solution corresponds to the original value of x_i. Naumann [Nau99], Lange [Lan04b], and Silverberg [Sil05] all follow this approach. The drawback of such an approach is that the equation has to be solved twice: during compression in order to determine ν, and again during decompression in order to determine x_i. Alternatively, one may choose to identify all points obtained from decompression, but it is not clear that the equivalence classes this gives are compatible with the group operation.

Naumann [Nau99, Chapter 5.2.3] suggests to use equation (3.7)

$$(3x_0^2 + 3\mu x_1 x_2 + A)^2 = 12x_0(x_0^3 + \mu x_1^3 + \mu^2 x_2^3 + Ax_0 + B),$$

which he obtains by eliminating the indeterminates y_0, y_1, y_2 from system (3.3), for an efficient representation of points $(X, Y) = (X_0, X_1, X_2, Y_0, Y_1, Y_2) \in T_3 = V_3(\mathbb{F}_q)$ via the coordinates (X_1, X_2, ν, λ). The equation is used to recover X_0 from X_1, X_2, where the number $0 \le \nu \le 3$ indicates which of the solutions corresponds to X_0, using some fixed ordering of $\mathbb{F}_q$. The curve equation is then used to recompute Y, and λ determines its sign. The numbers ν and λ can be saved in a total of 3 bits.

Another slightly more efficient approach would be to represent (X, Y) via (X_0, X_1, ν, λ) or (X_0, X_2, ν, λ), since the polynomial has only degree 3 in x_1, x_2. In this case, we would have $0 \le \nu \le 2$, and ν and λ together would still take 3 bits.

We derived in Chapter 3.3 a single equation whose $\mathbb{F}_q$-solutions describe the $\mathbb{F}_q$-points of the trace zero variety, up to a few well-described exceptions (see Lemma 3.10 and Proposition 3.11). Namely, this is equation (3.8)

$$\tilde{f}_n(x_0, \dots, x_{n-1}) = 0,$$

the Weil restriction of the n-th Semaev polynomial. This may be seen as a generalization of the equation of Naumann and can therefore be used for an efficient representation for the points of T_n in the same spirit: The compression algorithm drops one x_i in order to obtain a compact representation, and the decompression algorithm uses $\tilde{f}_n$ to recompute the missing coordinate. However, since $\tilde{f}_n$ has relatively large degree, this would identify more points than desired (3 for $n = 3$ and 31 for $n = 5$). Moreover, the computation of the Weil restriction of the Semaev polynomials requires a large amount of memory. It is already very demanding for $n = 5$. We present a different approach to the problem in the next section.

4.2 A representation from the symmetrized Semaev polynomial

We propose a compact representation of the points of $T_n = V_n(\mathbb{F}_q)$ using the equations derived in Chapter 3.4. We recall them briefly. The equation (3.11)

$$g_n(s_1, \dots, s_n) = 0$$

describes the points of T_n via the relations (3.12)

$$s_i = \tilde{e}_i(x_0, \dots, x_{n-1}), \quad i = 1, \dots, n.$$

Here g_n is the symmetrized n-th Semaev polynomial, and its degree is much lower than that of the Weil restricted Semaev polynomial $\tilde{f}_n$. The polynomials $\tilde{e}_i$ are obtained by Weil descent from the symmetric functions in the q-powers of x. All equations have coefficients in $\mathbb{F}_q$, and hence for a point $P = (X_0, \dots, X_{n-1}, Y_0, \dots, Y_{n-1}) \in V_n(\mathbb{F}_q)$ we have

$$S_i = \tilde{e}_i(X_0, \dots, X_{n-1}) \in \mathbb{F}_q, \quad i = 1, \dots, n.$$

Therefore, we may define the representation

$$\begin{array}{rcl} \mathcal{R} : V_n(\mathbb{F}_q) \setminus \{0\} & \longrightarrow & \mathbb{F}_q^{n-1} \\ (X_0, \dots, Y_{n-1}) & \longmapsto & (\tilde{e}_i(X_0, \dots, X_{n-1}))_{i=1}^{n-1}. \end{array}$$

It is clear that $\mathcal{R}$ has optimal size, since $|T_n| \approx q^{n-1}$.

Remark 4.3. Mapping to the vector $(\tilde{e}_i(X_0, \dots, X_{n-1}))$ with an arbitrary, fixed $\tilde{e}_i$ omitted is obviously an optimal representation. However, as we will see in the next section, the decompression algorithm recovers points with the smallest possible ambiguity when i is chosen such that the degree of g_n in s_i is minimal. One such choice is $i = n$, and therefore we choose to omit $\tilde{e}_n$.

4.3 Compression and decompression algorithms

In Algorithms 4.1 and 4.2 we show how to compute the representation via a compression algorithm and how to recover the full point (up to some small ambiguity) via a decompression algorithm. Afterwards we discuss how many points are identified by this representation.

Algorithm 4.1 Compression, $n \geq 3$

```
Input: $P = (X_0, \dots, X_{n-1}, Y_0, \dots, Y_{n-1}) \in V_n(\mathbb{F}_q)$
Output: Representation $(S_1, \dots, S_{n-1}) \in \mathbb{F}_q^{n-1}$ of $P$
1: for $i = 1, \dots, n-1$ do
2:     $S_i \leftarrow \tilde{e}_i(X_0, \dots, X_{n-1})$
3: end for
4: return $(S_1, \dots, S_{n-1})$
```

Remark 4.4. Because of Lemma 3.10, in line 10 of the decompression algorithm, for each solution X of system (4.1) one needs to check that the point $(X, Y) \in T_n$. This step can in practice be eliminated for $n = 3, 5$, as discussed in Remark 3.12.

For a small set of points, equation (3.11) vanishes when evaluated in the given numbers $S_1, \dots, S_{n-1}$. For such points P, any $t \in \mathbb{F}_q$ solves the equation $g_n(S_1, \dots, S_{n-1}, t) = 0$, making the computational effort for decompressing Compress(P) very large. Therefore, our decompression algorithm is not practical for such points. However, for almost all points $P \in V_n(\mathbb{F}_q)$ the polynomial $g_n(S_1, \dots, S_{n-1}, t)$ has only a small number of roots in t (upper bounded by the degree of g_n in the variable t). For our analysis, we assume that we are in the latter case. We have $P \in$ Decompress(Compress(P)), since the points of $V_n(\mathbb{F}_q)$ are described by $g_n(\tilde{e}_1(x_0, \dots, x_{n-1}), \dots, \tilde{e}_n(x_0, \dots, x_{n-1}))$. The relevant question is how many more points the output may contain.

Algorithm 4.2 Decompression, $n \geq 3$

```
Input: $(S_1, \ldots, S_{n-1}) \in \mathbb{F}_q{}^{n-1}$
Output: All points of $T_n = V_n(\mathbb{F}_q)$ that have $(S_1, \ldots, S_{n-1})$ as compact representation
 1: $L \leftarrow$ empty list
 2: $T \leftarrow$ set of solutions of $g_n(S_1, \ldots, S_{n-1}, t) = 0$ in $t$
 3: for each $\tau \in T$ do
 4:     find one solution $(X_0, \ldots, X_{n-1})$, if it exists, of the system
```

$$\begin{array}{rcl} S_1 & = & \tilde{e}_1(x_0, \ldots, x_{n-1}) \\ & \vdots & \\ S_{n-1} & = & \tilde{e}_{n-1}(x_0, \ldots, x_{n-1}) \\ \tau & = & \tilde{e}_n(x_0, \ldots, x_{n-1}) \end{array} \tag{4.1}$$

```
 5:     if no solution exists then
 6:         continue with the next $\tau$
 7:     end if
 8:     $X \leftarrow X_0 + \ldots + X_{n-1}\zeta^{n-1}$
 9:     $Y \leftarrow$ a $y$-coordinate corresponding to $X$, computed from the curve equation
10:     if $(X, Y) \in T_n$ then
11:         append $\pm P = (X, \pm Y)$ and all their Frobenius conjugates to $L$
12:     end if
13: end for
14: return $L$
```

First of all, by compressing a point, we lose the ability to distinguish between Frobenius conjugates of points, since for each solution of system (4.1), all Frobenius conjugates are also solutions. This can be compared to the fact that when using the "standard" compression, we lose the ability to distinguish between a point and its negative. If desired, a few extra bits can be used to remember that information. Alternatively, we can think of working in T_n modulo an equivalence relation that identifies the Frobenius conjugates of each point and its negative. This reduces the size of the group T_n by a factor $2n$, which is a small price to pay considering the amount of memory saved by applying the compression, especially since n is small in practice. In addition, this corresponds to the loss in Pollard–Rho security, since the algorithm can work in equivalence classes (see [DGM99]). Notice also that it is enough to compute one solution of system (4.1), since the set of all solutions consists precisely of the Frobenius conjugates of one point. This is because any polynomial in n variables which is left invariant by any permutation of the variables can be written uniquely as a polynomial in the elementary symmetric functions $e_1, \ldots, e_n$.

Now, how many different equivalence classes of points can be output by the decompression algorithm depends only on the degree of g_n in the last indeterminate. For $n = 3$ the degree is one and decompression therefore outputs only a single class. As n grows, the degree of the Semaev polynomial also grows, thus producing more ambiguity in the recovery process. This also reflects the growth in the number of extra points which satisfy the equation coming from the Semaev polynomial, as seen in Lemma 3.10.

Notice moreover that there may be solutions τ of $g_n(S_1, \ldots, S_{n-1}, t) = 0$ for which system (4.1) has no solutions, and that not all the solutions of system (4.1) produce an equivalence class of points on the trace zero variety. For example, if $X \in \mathbb{F}_{q^n}$ satisfies $f_n(X, X^q, \ldots, X^{q^{n-1}}) = 0$, the corresponding point $P = (X, Y) \in E$ may have $Y \in \mathbb{F}_{q^{2n}} \setminus \mathbb{F}_{q^n}$. In this case $P \notin T_n$.

Since our algorithms are most useful for $n = 3$ and 5, an asymptotic complexity analysis for general n does not make much sense. In fact, it is easy to count the number of additions, multiplications, and squarings in $\mathbb{F}_q$ needed to compute the representation just from looking at the formulas for $s_1, \ldots, s_{n-1}$. We do this for the cases $n = 3$ and 5 in Sections 4.5.1 and 4.5.2, respectively. There, we also discuss the efficiency of our decompression algorithm and how it compares to the approaches of [Nau99, Sil05].

4.4 Group operation

In order to compute with points of T_n, we suggest to decompress a point, perform the operation in $E(\mathbb{F}_{q^n})$, and compress again the result. Since compression and decompression is very efficient, this adds only little overhead. In an environment with little storage and/or bandwidth capacity, the memory savings of compressed points may well be worth this small trade-off with the efficiency of the arithmetic. Also notice that scalar multiplication of trace zero points in $E(\mathbb{F}_{q^n})$ is more efficient than scalar multiplication of arbitrary points of $E(\mathbb{F}_{q^n})$, due to a speed-up using the Frobenius endomorphism, as pointed out by Frey [Fre99] and studied in detail by Lange [Lan01, Lan04b] and subsequently by Avanzi and Cesena [AC07], see also [ACD$^+$06, Chapter 15.3.2].

Our recommendation corresponds to usual implementation practice in the setting of point compression: Even when a method to compute with compressed points is available, it is usually preferable to perform decompression, compute with the point in its original representation, and compress the result. For example, Galbraith and Lin show in [GL09] that although it is possible to compute pairings using the x-coordinates of the input points only, it is more efficient in most cases (namely, whenever the embedding degree is greater than 2) to recompute the y-coordinates of the input points and perform the pairing computation on the full input points. As a second example, let us consider the following two methods for scalar multiplication by k of an elliptic curve point $P = (X, Y)$ when only X is given:

(i) Use the Montgomery ladder, which computes the x-coordinate of $[k]P$ from X only.

(ii) Find Y by computing a square root, apply a fast scalar multiplication algorithm to (X,Y), and return only the x-coordinate of the result.

Most recent speed records for scalar multiplication on elliptic curves have been set using algorithms that need the full point P, in other words with the second approach, see e.g. [BDL$^+$12, LS12, OLAR13, FHLS13]. Timings typically ignore the additional cost for point decompression, but there is strong evidence that on a large class of elliptic curves the second approach is faster and therefore preferable whenever protection against side channel attacks (i.e. timing invariance) is not required. Moreover, whenever automorphisms are used to speed up scalar multiplication (e.g. GLV/GLS curves and the trace zero subgroup, see Section 2.4 for how to do this), point addition (and not just scalar multiplication) is needed, and therefore the Montgomery ladder cannot be applied directly. This is the basis for our suggestion to follow the second approach when working with compressed points of T_n.

Remark 4.5. When performing the group operation, it is not necessary to compute the entire equivalence class (i.e. all preimages) of a given point during decompression. Instead, it suffices to compute one arbitrary representative of the class, perform the scalar multiplication on this representative, and compress the result again. Since scalar multiplication commutes with both the Frobenius map and the negation of a point, the final result is independent of the choice of representative, as long as the representation identifies only one class of Frobenius conjugates for that particular point.

In Diffie–Hellman key exchange, the goal is to agree on a *unique* common secret, while when following our approach of computing with compressed coordinates that represent equivalence classes of points, two parties a priori only agree on a shared equivalence class instead of a single point. Since the compressed coordinates are a unique representation of the equivalence class, we suggest to apply a key derivation function directly to these coordinates. This means that an execution of the compression algorithm is necessary just before the key derivation function is applied.

Alternatively, one could fix an order on the elements of $\mathbb{F}_{q^n}^2$, choose the representative of the class which has the smallest coordinates with respect to this order, and then apply a key derivation function to this point. For this approach, it would be necessary to compute all elements of the equivalence class (and not just one representative).

4.5 Explicit equations and comparison with other representations

We present explicit equations and experimental results for $n = 3, 5$. All computations were done with Magma version 2.19.3 [BCP97], running on one core of an Intel Xeon X7550 Processor (2.00 GHz) on a Fujitsu Primergy RX900S1. Our Magma programs are straightforward implementations of the methods presented here and are only meant as an indication. No particular effort has been put into optimizing them.

We also give a detailed analysis of our compression and decompression algorithms for $n = 3, 5$. Where possible, we count squarings (S), multiplications (M), and inversions (I) in $\mathbb{F}_q$. We do not count multiplication by constants, since they can often be chosen small (see [Lan04b]), and multiplication can then be performed by repeated addition.

On the basis of this analysis, we compare our algorithms to those of Naumann [Nau99] for $n = 3$ and Silverberg [Sil05] for $n = 3, 5$.

4.5.1 Explicit equations and comparison for $n = 3$

We give explicit equations for $n = 3$, where $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - \mu)$ is a Kummer extension of $\mathbb{F}_q$. Some of them were already presented in Chapter 3.4.1. The symmetrized third Semaev polynomial is equation (3.13)

$$g_3(s_1, s_2, s_3) = s_2^2 - 4s_1s_3 - 4Bs_1 - 2As_2 + A^2$$

and describes the trace zero subgroup via system (3.14)

$$\begin{array}{rclcl} s_1 & = & x + x^q + x^{q^2} & = & 3x_0 \\ s_2 & = & x^{1+q} + x^{1+q^2} + x^{q+q^2} & = & 3x_0^2 - 3\mu x_1x_2 \\ s_3 & = & x^{1+q+q^2} & = & x_0^3 - 3\mu x_0x_1x_2 + \mu x_1^3 + \mu^2 x_2^3. \end{array}$$

So for compression of a point $(x_0, x_1, x_2, y_0, y_1, y_2)$, we use the coordinates

$$(s_1, s_2) = (3x_0, 3x_0^2 - 3\mu x_1x_2),$$

and for decompression, we have to solve $g_3(s_1, s_2, s_3) = 0$ for s_3, where g_3 is given by equation (3.13). Since the equation is linear in s_3, the missing coordinate can be recovered uniquely, except when $s_1 = 0$. This is the case only for a small set of points. Notice

Table 4.1 Average time in milliseconds for compression/decompression of one point, $n = 3$, smallest values in italics

q	$2^{10}-3$	$2^{20}-3$	$2^{40}-87$	$2^{60}-93$	$2^{79}-67$
Compression s_i	0.007	0.014	0.028	0.039	0.064
Compression t_i	*0.002*	*0.007*	*0.008*	*0.010*	*0.015*
Decompression s_i	0.124	0.159	0.731	0.987	1.586
Decompression t_i	*0.090*	*0.132*	*0.610*	*0.956*	*1.545*

moreover that the points $(0, s_2, s_3)$ with $s_2^2 - 2As_2 + A^2 = 0$ satisfy equation (3.13) for every s_3. The only ambiguity in decompression comes from solving system (3.14), which yields the Frobenius conjugates x, x^q, x^{q^2} of the original x. So for $n = 3$ this gives an optimal representation in our sense.

The following representation is equivalent to the above but easier to compute. Set

$$t_1 = x_0,\ t_2 = x_1x_2,\ t_3 = x_1^3 + \mu x_2^3, \tag{4.2}$$

and take (t_1, t_2) as a representation. The relation between the two sets of coordinates is

$$s_1 = 3t_1,\ s_2 = 3t_1^2 - 3\mu t_2,\ s_3 = t_1^3 - 3\mu t_1t_2 + \mu t_3.$$

In this case, we recover t_3 from the equation

$$-3t_1^4 + 18\mu t_1^2t_2 + 9\mu^2t_2^2 - 12\mu t_1t_3 - 12Bt_1 - 6At_1^2 + 6A\mu t_2 + A^2 = 0.$$

The equation is linear in t_3, thus making point recovery unique whenever $t_1 \neq 0$, but the total degree is higher. Compared to the representation (s_1, s_2), fewer operations are needed for compression and for computing the solutions of the system during decompression. Thus, compression and decompression for this variant of the representation are more efficient. We give timings for 10, 20, 40, 60, and 79 bit fields in Table 4.1. The smallest values for compression and decompression respectively are emphasized in italics, and we see that compression is about a factor 3 to 4 faster and decompression is slightly faster for the second method. Notice that decompression timings are for recomputing the x-coordinate only.

Example 4.6. Let E be the curve $y^2 = x^3 + x + 368$ over $\mathbb{F}_q$, where $q = 2^{79} - 67$ is a 79-bit prime and $\mu = 3$. The trace zero subgroup of $E(\mathbb{F}_{q^3})$ has prime order of 158 bits. We choose a random point in T_3 (to save some space, we write only x-coordinates)

$$P = 260970034280824124824722 + 431820813779055023676698\zeta + 496444425404915392572065\zeta^2$$

and compute

$$\begin{aligned}
&\mathrm{Compress}(P) = (178447193035157787121145, 159414355696879147312583)\\
&\mathrm{Decompress}(178447193035157787121145, 159414355696879147312583) =\\
&\quad\{260970034280824124824722 + 431820813779055023676698\zeta + 496444425404915392572065\zeta^2,\\
&\quad\ 260970034280824124824722 + 318397306102476549147695\zeta + 124410673032925784958936\zeta^2,\\
&\quad\ 260970034280824124824722 + 458707699733097601881649\zeta + 880707211767879971750 41\zeta^2\}
\end{aligned}$$

where the results of decompression are exactly the Frobenius conjugates of P. In our Magma implementation, we solve system (3.14) over $\mathbb{F}_q$ similarly to how one would do it by hand, as described below. Notice that the solutions could also be found by computing the roots of the polynomial $x^3 - s_1x^2 + s_2x - s_3$ over $\mathbb{F}_{q^3}$, but since the system is so simple for $n = 3$, solving it directly is faster in all instances.

When using the second variant of the representation, we compute

$$(t_1, t_2) = (26097003428082412482472 2, 4927210325282564313084 37)$$

and naturally get the same result for decompression by solving system (4.2) in a similar way.

Operation count for representation in the s_i. Compressing a point clearly takes 1S+1M. Decompression requires the following steps.

- Evaluating $g_3(s_1, s_2, s_3)$ in the first two indeterminates and solving for the third indeterminate means computing $s_3 = \frac{1}{4s_1}(s_2(s_2 - 2A) - 4Bs_1 + A^2)$, which takes 2M+1I.
- Given s_1, s_2, s_3, we need to solve system (3.14) for x, or for x_0, x_1, x_2. The most obvious way would be to compute the roots of the univariate polynomial $x^3 - s_1x^2 + s_2x - s_3$ over $\mathbb{F}_{q^3}$. Finding all roots of a degree d polynomial over $\mathbb{F}_{q^n}$ takes $O(n^{\log_2 3}d^{\log_2 3} \log d \log(dq^n))$ operations in $\mathbb{F}_q$ using Karatsuba's algorithm for polynomial multiplication (see [GvzG99]). In our case, the degree and n are constants, and hence factoring this polynomial takes $O(\log q)$ operations in $\mathbb{F}_q$. However, since the system is so simple, in practice it is better to solve directly for x_0, x_1, x_2 over $\mathbb{F}_q$. We know that the system has exactly three solutions (except in very few cases, where it has a unique solution in $\mathbb{F}_q$, i.e. $x_1 = x_2 = 0$). We get x_0 from s_1 for free. Assuming that $x_1 \neq 0$ (the special case when $x_0 = 0$ is easier than this general case), we can solve the second equation for x_2, plug this into the third equation, and multiply by the common denominator $27\mu^3x_1^3$. In this way, we obtain the equation
$$27\mu^4x_1^6 + 27\mu^3(x_0(s_2 - 2x_0^2) - s_3)x_1^3 + \mu^2(3x_0^2 - s_2)^3 = 0,$$
which must be solved for x_1. The coefficient of x_1^6 is a constant, the coefficient of x_1^3 can be computed with 1S+1M, and the constant term can then be computed with 1S+1M. Now we can solve for x_1^3 with the quadratic formula, which takes 1S and a square root in $\mathbb{F}_q$ for the first value, which will have either no or three distinct cube roots. In case it has none, we compute the second value for x_1^3, using only an extra addition, and the three distinct cube roots of this number. This gives a total of 3 values for x_1. Finally, we can compute $x_2 = \frac{3x_0^2 - s_2}{3\mu x_1}$, which takes 1M+1I for the first, and a multiplication by the inverse of a cube root of unity for the other two values. Altogether, solving system (3.14) takes a total of at most 3S+3M+1I, 1 square root, and 2 cube roots in $\mathbb{F}_q$.
- Finally, for each of the at most 3 values for x, we recompute a corresponding y-coordinate from the curve equation and check that it belongs to $\mathbb{F}_{q^3}$. Since these are standard procedures for elliptic curves, we do not count operations for these tasks.

Therefore, the decompression algorithm takes at most 3S+5M+2I, one square root, and two cube roots in $\mathbb{F}_q$. The cost of computing the roots depends on the specific choice of the field and on the implementation, but it clearly dominates this computation.

Operation count for representation in the t_i. In this case, compression takes only 1M. For decompression, we proceed as follows.

- Given t_1 and t_2, we recover t_3 from the equation $t_3 = \frac{1}{12\mu t_1}(-3t_1^4 + (18\mu t_1^2 + 9\mu^2 t_2 + 6A\mu)t_2 - 12Bt_1 - 6At_1^2 + A^2)$. This takes 2S+2M+1I.

- To solve system (4.2), again assuming $x_1 \neq 0$, we have to find the roots of the equation
$$x_1^6 - t_3 x_1^3 + \mu t_2^3 = 0.$$
The coefficients of this equation can be computed with a total of 1S+1M. We proceed as above to compute 3 values for x_1 using 1S, 1 square root, and 2 cube roots. Finally, we compute $x_2 = \frac{t_2}{x_1}$ using 1M+1I. Thus, solving the system takes a total of at most 2S+2M+1I, 1 square root, and 2 cube roots.

In total, decompression takes at most 4S+4M+2I, 1 square root, and 2 cube roots. The cost of this computation is comparable to the decompression using s_i. This corresponds to our experimental results with Magma (see Table 4.1).

Comparison with Silverberg's method. The representation of [Sil05] consists of the last $n-1$ Weil restriction coordinates, together with three extra bits, say $0 \leq \nu \leq 3$ to resolve ambiguity in recovering the x-coordinate and $0 \leq \lambda \leq 1$ to determine the sign of the y-coordinate. So in our notation, Silverberg's representation of a point $(x, y) \in T_3$ is via the coordinates (x_1, x_2, ν, λ). The compression and decompression algorithms (in characteristic not equal to 3) carry out essentially the same steps:

- Compute a univariate polynomial of degree 4. The coefficients are polynomials over $\mathbb{F}_q$ in 2 indeterminates of degree at most 4.

- Compute the (up to 4) roots of this polynomial. During compression, this determines ν. During decompression, ν determines which root is the correct one, and it is then used to compute x_0 via addition and multiplication with constants.

- During decompression, compute the y-coordinate from the curve equation, using λ to determine its sign. As above, we disregard this step.

Since [Sil05] does not contain a detailed analysis of the decompression algorithm, we cannot compare the exact number of operations. However, the essential difference with our approach is that Silverberg's compression and decompression algorithms both require computing the roots of a degree 4 polynomial over $\mathbb{F}_q$. For compression, this is clearly more expensive than our method, which consists only of evaluating some small expressions. For decompression, this is also less efficient than our method, which computes only a root of a quadratic polynomial, since running a root finding algorithm, or using explicit formulas for the solutions (i.e. solving the quartic by radicals), is much more complicated than computing the roots of our equation.

One might argue that it is possible to represent (x, y) via the coordinates (x_1, x_2). In such a case, compression would consist simply of dropping y and x_0 and would therefore have no computational cost. Without remembering ν and λ to resolve ambiguity, this representation would identify up to 4 x-coordinates and up to 8 full points. This is not much worse than our representation, which identifies up to 3 x-coordinates and 6 full points. However, it is not clear that this identification is compatible with scalar multiplication of points. Therefore, one may want to use at least ν to distinguish between the recovered x-coordinates. This is in contrast with our situation, where we know exactly which points are recovered during decompression (i.e. the three Frobenius conjugates of the original point).

Identifying these three points is compatible with scalar multiplication, since $P = \varphi^i(Q)$ implies $[k]P = \varphi^i([k]Q)$ for all $k \in \mathbb{N}$ and $P, Q \in T_3$, and so no extra bits are necessary.

Comparison with Naumann's method. Naumann's method, as explained in Section 4.1, is analogous to the one of Silverberg. He drops x_0 and uses a quartic equation to recompute it. His equation is a different one, yet the analysis of his method is analogous to that of Silverberg's method, and the conclusions are the same. In particular, his algorithms are less efficient than ours, and it is not clear whether it is possible to drop ν from the representation and still have a well-defined scalar multiplication.

4.5.2 Explicit equations and comparison for $n = 5$

As explained in Chapter 3.4.2, the symmetrized fifth Semaev polynomial is too big to be printed here, but a computer program can easily work with it. It has total degree 8 and degree 6 in the last indeterminate.

The fact that we recover the missing coordinate from a degree 6 polynomial introduces some indeterminacy in the decompression process. However, extensive Magma experiments for different field sizes and curves show that for more than 90% of all points in T_5, only a single class of Frobenius conjugates is recovered. For another 9%, two classes (corresponding to 10 x-coordinates) are recovered. Thus the ambiguity is very small for a great majority of points. In any case, this improves upon the approach of [Sil05], where the missing coordinate is recovered from a degree 27 polynomial, thus possibly yielding 27 different x-coordinates.

The Weil restriction of the symmetric functions is given in Section 3.4.2. The compression algorithm computes $s_1, \dots, s_4$ according to these formulas over $\mathbb{F}_q$. The decompression algorithm solves a degree 6 equation for s_5 and then recomputes the x-coordinate of the point. For the last step, we test two methods: We compute x by factoring the polynomial $x^5 - s_1x^4 + s_2x^3 - s_3x^2 + s_4x - s_5$ over $\mathbb{F}_{q^5}$, and we compute $x_0, \dots, x_4$ by solving the system over $\mathbb{F}_q$ with a Gröbner basis computation. Our experiments show that polynomial factorization can be up to 20 times as fast as computing a lexicographic Gröbner basis in Magma for some choices of q, and the entire decompression algorithm can be up to a factor 6 faster when implementing the polynomial factorization method. We give some exemplary timings for both methods for fields of 10, 20, 30, 40, 50 and 60 bits in Table 4.2, where as before the smallest values are printed in italics. However, these experimental results can only be an indication. In Magma, the performance of the algorithms depends on the specific choice of q. In addition, any implementation exploiting a special shape of q would most likely produce better results.

As for $n = 3$, we suggest an equivalent representation (t_1, t_2, t_3, t_4) where

$$
\begin{array}{rcl}
t_1 & = & x_0 \\
t_2 & = & x_1x_4 + x_2x_3 \\
t_3 & = & x_1^2x_3 + x_1x_2^2 + \mu x_3^2x_4 + \mu x_2x_4^2 \\
t_4 & = & \mu x_2^2x_3^2 + \mu x_1^2x_4^2 - \mu x_1x_3^3 - x_1^3x_2 - \mu x_2^3x_4 - \mu^2x_3x_4^3 + \mu x_1x_2x_3x_4 \\
t_5 & = & x_1^5 + \mu x_2^5 + \mu^2x_3^5 + \mu^3x_4^5 + 5\mu x_1^2x_2x_3^2 + 5\mu x_1^2x_2^2x_4 + 5\mu^2x_2^2x_3x_4^2 \\
 & & +5\mu^2x_1x_3^2x_4^2 - 5\mu x_1^3x_3x_4 - 5\mu^2x_2x_3^3x_4 - 5\mu^2x_1x_2x_4^3 - 5\mu x_1x_2^3x_3
\end{array}
\tag{4.3}
$$

and

$$
\begin{array}{rcl}
s_1 & = & 5t_1 \\
s_2 & = & 10t_1^2 - 5\mu t_2 \\
s_3 & = & 10t_1^3 - 15\mu t_1t_2 + 5\mu t_3 \\
s_4 & = & 5t_1^4 - 15\mu t_1^2t_2 + 10\mu t_1t_3 + 5\mu t_4 \\
s_5 & = & t_1^5 - 5\mu t_1^3t_2 + 5\mu t_1^2t_3 + 5\mu t_1t_4 + \mu t_5.
\end{array}
\tag{4.4}
$$

Table 4.2 Average time in milliseconds for compression/decompression of one point, $n = 5$, smallest values in italics

q	$2^{10}-3$	$2^{20}-5$	$2^{30}-173$	$2^{40}-195$	$2^{50}-113$	$2^{60}-695$
Compression s_i	0.041	0.048	0.052	0.106	0.108	0.112
Compression t_i	*0.017*	*0.022*	*0.024*	*0.031*	*0.021*	*0.048*
Decompression s_i poly factorization	*5.536*	*16.480*	*21.423*	*45.080*	*55.872*	*59.520*
Decompression s_i Gröbner basis	24.134	26.470	39.593	101.559	104.490	118.991
Decompression t_i Gröbner basis	38.375	40.198	60.438	132.484	133.088	150.083

Compared to the representation in the s_i, this representation gives a faster compression but a slower decompression. Therefore, this approach may be useful in a setting where compression must be particularly efficient.

For decompression, the missing coordinate t_5 can be recomputed from a degree 6 equation, which we obtain by substituting the relations (4.4) into the symmetrized fifth Semaev polynomial. Afterwards we may either recompute $s_1, \dots, s_5$ from $t_1, \dots, t_5$ according to system (4.4) and solve $x^5 - s_1x^4 + s_2x^3 - s_3x^2 + s_4x - s_5$ for x, or else we may solve system (4.3) directly for $x_0, \dots, x_4$ with Gröbner basis techniques. The polynomial factorization method is equivalent to using the representation in the s_i, only that some of the computations are shifted from the compression to the decompression algorithm. The Gröbner basis method (use t_i and compute Gröbner basis, "second method") compares to using s_i with Gröbner basis ("first method") as given in Table 4.2. We see that the second method is a factor 2 to 3 faster in compression but slower in decompression. The reason for this is that the polynomial used to recompute the missing coordinate is more complicated for the second method and evaluation of polynomials is quite slow in Magma. Solving for the missing coordinate takes 5 times longer for the second method. The solution of system (4.1), which we achieve by computing a lexicographic Gröbner basis and solving the resulting triangular system in the obvious way, takes the same amount of time in both cases.

We now give an example of our compression/decompression algorithms including two points P on the trace zero variety where Decompress(Compress(P)) produces the minimum and maximum possible number of outputs.

Example 4.7. Let E be the curve $y^2 = x^3 + x + 12$ over $\mathbb{F}_q$, where $q = 2^{40} - 195$ is a 40-bit prime, and $\mu = 2$. The trace zero subgroup of $E(\mathbb{F}_{q^5})$ has prime order of 160 bits. We choose a random point in T_5

$$P = 825957898670 + 728788139043\zeta + 852868227354\zeta^2 + 795116698017\zeta^3 + 343259616641\zeta^4$$

and compute

$$\begin{aligned}
&\mathrm{Compress}(P) = (831254610607, 6785365592, 993014138866, 657582814119)\\
&\mathrm{Decompress}(831254610607, 6785365592, 993014138866, 657582814119) =\\
&\quad \{825957898670 + 728788139043\zeta + 852868227354\zeta^2 + 795116698017\zeta^3 + 343259616641\zeta^4,\\
&\quad 825957898670 + 993461058450\zeta + 986548584673\zeta^2 + 439026061373\zeta^3 + 212468827743\zeta^4,\\
&\quad 825957898670 + 114573116312\zeta + 987359757846\zeta^2 + 813883702713\zeta^3 + 872169726781\zeta^4,\\
&\quad 825957898670 + 866712637108\zeta + 837469697186\zeta^2 + 806866513841\zeta^3 + 685256896367\zeta^4,\\
&\quad 825957898670 + 728788139043\zeta + 852868227354\zeta^2 + 795116698017\zeta^3 + 343259616641\zeta^4\}.
\end{aligned}$$

When using the second variant of the representation, we compute

$$(t_1, t_2, t_3, t_4) = (825957898670, 1041513998836, 427081277156, 473899514349).$$

For this point, the results of decompression are exactly the Frobenius conjugates of P. However, this is not always the case. In rare cases, the algorithm may recover up to six classes of Frobenius conjugates. In a one week search, we were able to find a point for which this happens:

$$P = 365893271595 + 7683805409256\zeta + 1600452893836\zeta^2 + 935066335578\zeta^3 + 312458428681\zeta^4.$$

Operation count for representation in the s_i. Given $x_0, \dots, x_4$, the numbers $t_1, \dots, t_4$ can be computed with a total of 5S+13M according to (4.3). Then $s_1, \dots, s_4$ can be computed from those numbers with 2S+3M as given in (4.4). This seems to be the best way to compute $s_1, \dots, s_4$, since these formulas group the terms that appear several times. Hence compression takes a total of 7S+16M.

For decompression, the most costly part of the algorithm is factoring the polynomials. First, the algorithm has to factor a degree 6 polynomial over $\mathbb{F}_q$, and next, a degree 5 polynomial over $\mathbb{F}_{q^5}$. The asymptotic complexity for both of these is $O(\log q)$ operations in $\mathbb{F}_q$.

Operation count for representation in the t_i. Compression takes 5S+13M. For decompression, we can either recompute $s_1, \dots, s_5$ from $t_1, \dots, t_5$ and factor the polynomial, in which case this approach is exactly the same as the above. Or else we can solve system (4.3) by means of a Gröbner basis computation over $\mathbb{F}_q$. Since there are no practically meaningful bounds for Gröbner basis computations, a complexity analysis of this approach makes no sense.

Comparison with Silverberg's method. Concrete equations are presented in [Sil05] for the case where the ground field has characteristic 3. The most costly parts of the algorithms are computing the resultant of two polynomials of degree 6 and 8 with coefficients in $\mathbb{F}_q$ and finding the roots of a degree 27 polynomial over $\mathbb{F}_q$. In general, resultant computations are difficult, and the polynomial to be factored has much larger degree than those in our algorithm. In Silverberg's approach, five extra bits are required to distinguish between the possible 27 roots of the polynomial.

Although neither Silverberg nor we give explicit equations for larger n, our understanding is that our algorithm scales better with increasing n, since our method is more natural and respects the structure of the group.

4.6 Conclusions

The Semaev polynomials give rise to a useful equation describing the $\mathbb{F}_q$-rational points of the trace zero variety. Its significance is that it is one single equation in the x-coordinates of the elliptic curve points, but unfortunately its degree grows quickly with n. Using this equation, we obtain an efficient method of point compression and decompression. It computes a representation for the $\mathbb{F}_q$-points of the trace zero variety that is optimal for $n = 3$ and optimal in practice $n = 5$. Our polynomials have lower degree than those used in the representations of [Sil05] (1 compared to 4 for $n = 3$, and 6 compared to 27 for $n = 5$) and [Nau99] (1 compared to 4 for $n = 3$), thus allowing more efficient decompression and less ambiguity in the recovery process. Finally, our representation is interesting from a mathematical point of view, since it is the first representation (to our knowledge) that is compatible with scalar multiplication of points.

Chapter 5

An optimal representation via rational functions

In this chapter we propose a new optimal-size representation for the elements of the trace zero subgroup associated to an elliptic or hyperelliptic curve of any genus g and any field extension of prime degree n. It is conceptually different from all previous representations, including the one of Chapter 4 of this thesis, and it is the first representation that works for elliptic curves with $n > 5$, for hyperelliptic curves of genus 2 with $n > 3$, and for hyperelliptic curves of genus $g > 2$. The basic idea is to represent a given divisor class via the coefficients of the rational function whose associated principal divisor is the trace of the given divisor. Our representation enjoys convenient properties, for example it identifies well-defined equivalence classes of points, and scalar multiplication is well-defined on such classes. In the context of many DLP-based cryptosystems such as Diffie–Hellman, where the only operation required is scalar multiplication of points, this enables us to compute with equivalence classes of trace zero elements, and no extra bits are required to distinguish between the different representatives.

Moreover, we give a compression algorithm to compute the representation of a given point and a decompression algorithm to compute (a representative of the equivalence class of) the original point. We show that these algorithms are comparably or more efficient than all previously known compression and decompression methods and that they are efficient even for medium to large values of g and n.

Assumptions and Notation. *In this chapter, let C be a projective elliptic or hyperelliptic curve of genus g defined over a finite field $\mathbb{F}_q$ of odd characteristic and given by an affine equation of the form*

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

where $f \in \mathbb{F}_q[x]$ is monic of degree $2g+1$ and has no multiple zeros. We assume that $\mathbb{F}_q$ does not have characteristic 2 only for convenience and ease of exposition, all our statements carry over to the characteristic 2 case with some minor adjustments. See also Remark 5.4.

Roadmap. In this chapter, we give a unified treatment of the new representation for elliptic and hyperelliptic curves, using the notation introduced in Chapter 2.2. In Chapter 6 we then specialize these results to elliptic curves, where a number of simplifications can be made. This chapter is organized as follows. In Section 5.1 we discuss the representation. We explain how to compute the rational function in Section 5.2 and give full compression and decompression algorithms in Section 5.3. We study explicit equations for the case $g = 2, n = 3$ in Section 5.4 and present some implementation results, as well as a detailed comparison with the other compression methods, in Section 5.5. Explicit equations for $g = 1, n = 3, 5$ are postponed to Chapter 6.

5.1 An optimal representation via rational functions

We propose an optimal representation for the trace zero variety. Our representation relies on the observation that for every $[D] \in T_n$ there is a rational function h_D on C with

$$\mathrm{Tr}(D) = \mathrm{div}(h_D).$$

In this section we discuss how to represent h_D via $g(n-1)$ elements of $\mathbb{F}_q$. Such a representation is optimal, since $|T_n| \approx q^{g(n-1)}$.

A rather trivial example is the case of elliptic curves E and extension degree $n = 2$, where

$$T_2 = \{(X,Y) \in E(\mathbb{F}_{q^2}) \mid X \in \mathbb{F}_q, Y \in (\mathbb{F}_{q^2} \setminus \mathbb{F}_q) \cup \{0\}\} \cup \{\mathcal{O}\}.$$

In particular, if $D = (X,Y) - \mathcal{O}$, then $h_D = x - X$, and the x-coordinate of the points of T_2 yields an optimal representation (see Proposition 3.2, Corollary 4.2). This statement can be generalized to higher genus curves when $n = 2$: The u-polynomial of the Mumford representation yields an optimal-size representation.

Proposition 5.1. *Let C be an elliptic or hyperelliptic curve of genus $g \geq 1$ defined over $\mathbb{F}_q$, and let $T_2 \subseteq \mathrm{Pic}^0_C(\mathbb{F}_{q^2})$ be the trace zero subgroup corresponding to the field extension $\mathbb{F}_{q^2}|\mathbb{F}_q$. Then for every reduced divisor $D = [u,v]$ with $[D] \in T_2$ we have $u \in \mathbb{F}_q[x]$. Therefore, the map*

$$\begin{array}{cccc} \mathcal{R}: & T_2 & \longrightarrow & \mathbb{F}_q[x]_{\leq g} \\ & [u,v] & \longmapsto & u \end{array}$$

yields an optimal representation for the elements of T_2, where $\mathbb{F}_q[x]_{\leq g}$ denotes the polynomials of degree smaller than or equal to g with coefficients in $\mathbb{F}_q$. This representation has length $g \log_2 q + 1$, and it identifies every divisor class $[D]$ with its conjugate $[w(D)] = [\varphi(D)]$.

This proposition is a special case of Theorem 5.3, but a direct proof is very easy:

Proof. Let D be a reduced divisor with Mumford representation $[u,v]$. Assume $[D] \in T_2$, or equivalently $D \sim w(\varphi(D))$. Since D is reduced, $w(\varphi(D))$ is also reduced, hence $D = w(\varphi(D))$. Since the Mumford representation of $w(\varphi(D))$ is $[u^\varphi, -v^\varphi]$, we have $u = u^\varphi$ and therefore $u \in \mathbb{F}_q[x]$. □

Remark 5.2. Notice that if $g = 1$ and $D = (X,Y) - \mathcal{O}$, then $u(x) = x - X$ is a vertical line, from which (X,Y) can be recovered up to sign. Since $u(x)$ is determined by the coefficient X, the representation consists only of the x-coordinate of the point. Moreover, we do not need to include the extra bit in the representation, since we always have $r = 1$ for $D \neq 0$. Hence this representation coincides with the trivial optimal representation from Corollary 4.2.

We now proceed to solve the problem in the case when n is any prime. Let D be a reduced divisor. We propose to represent an element $[D]$ of T_n via the rational function h_D on C with divisor

$$\mathrm{div}(h_D) = \mathrm{Tr}(D).$$

Such a function is unique up to multiplication by a constant. We now establish some properties of h_D. In particular, we show that a normalized form of h_D can be represented via $g(n-1)$ elements of $\mathbb{F}_q$. This gives an optimal representation for the elements of T_n which identifies at most n^g divisor classes. After proving the theorem, we discuss how to compute the representation via a Miller-type algorithm (compression) and how to recover the original divisor or class of divisors (decompression).

Theorem 5.3. *Let $D = P_1 + \ldots + P_r - r\mathcal{O}$ be a reduced divisor such that $[D] \in T_n$, and let $h_D \in \mathbb{F}_q(C)$ be a function such that $\operatorname{div}(h_D) = \operatorname{Tr}(D)$. Write $D = D_1 + \ldots + D_t$, where D_i are reduced prime divisors defined over $\mathbb{F}_{q^n}$. Then:*

(i) *$h_D = h_{D,1}(x) + y h_{D,2}(x)$ with $h_{D,1}, h_{D,2} \in \mathbb{F}_q[x]$.*

(ii) *$H_D(x) := h_{D,1}(x)^2 - f(x)h_{D,2}(x)^2 \in \mathbb{F}_q[x]$ has degree rn, and its zeros over $\overline{\mathbb{F}}_q$ are exactly the x-coordinates of the points $\varphi^j(P_1), \ldots, \varphi^j(P_r)$ for $j = 0, \ldots, n-1$. Equivalently, $H_D = N(u)$, where $D = [u, v]$.*

(iii) *$\deg h_{D,1} \leq \lfloor \frac{nr}{2} \rfloor$ and $\deg h_{D,2} \leq \lfloor \frac{nr-2g-1}{2} \rfloor$, where equality holds for the degree of $h_{D,1}$ if r is even or $n = 2$ and equality holds for the degree of $h_{D,2}$ if r is odd and $n \neq 2$.*

(iv) *Up to multiplication by a constant, the function h_D is uniquely determined by $g(n-1)$ elements of $\mathbb{F}_q$ and $\delta \in \mathbb{F}_2$, where $\delta = 1$ if and only if $r = g$. If $g = 1$, then we always have $r = g$, so we do not need to store $\delta \in \mathbb{F}_2$ in the representation.*

(v) *Let F be a reduced divisor. Then $h_D = h_F \in \mathbb{F}_q(C)$ if and only if F is of the form $F = \varphi^{j_1}(D_1) + \ldots + \varphi^{j_t}(D_t)$ for some $0 \leq j_1, \ldots, j_t \leq n-1$. In particular, there are at most n^g reduced divisors F such that $h_F = h_D$.*

(vi) *Let $D_i = [u_i, v_i]$ be a prime divisor, $i \in \{1, \ldots, t\}$. Then: $h_{D,2} \equiv 0 \bmod u_i$ if and only if $w(D_i) = \varphi^j(D_k)$ for some $j \in \{0, \ldots, n-1\}$ and some $k \in \{1, \ldots, t\}$ if and only if $\operatorname{Tr}(D_i) = w(\operatorname{Tr}(D_k))$. Suppose in addition that $n \neq 2$, then: $w(D_i) = \varphi^j(D_i)$ for some $j \in \{0, \ldots, n-1\}$ if and only if $D_i = w(D_i)$.*

(vii) *Let $n \neq 2$, let $D_i = [u_i, v_i] \leq D$ be a reduced prime divisor with $D_i \neq w(D_i)$, and let $\ell, m \geq 0$. Then we have $\operatorname{Tr}(D) = m \operatorname{Tr}(D_i) + \ell \operatorname{Tr}(w(D_i)) + \operatorname{Tr}(G)$ for some G, where $\operatorname{Tr}(D_i) \not\leq \operatorname{Tr}(G)$ and $\operatorname{Tr}(w(D_i)) \not\leq \operatorname{Tr}(G)$, if and only if $N(u_i)^{\min\{\ell,m\}}$ exactly divides h_D (as polynomials).*

Proof. Since $[D] \in T_n$, we have $0 \sim \operatorname{Tr}(D) \in \operatorname{Div}_C(\mathbb{F}_q)$. Hence there exists an $h_D \in \mathbb{F}_q(C)$ such that $\operatorname{div}(h_D) = \operatorname{Tr}(D)$. The function h_D is uniquely determined up to multiplication by a constant.

(i) The function h_D is a polynomial, since it has its only pole at $\mathcal{O}$. Using the curve equation $y^2 = f(x)$, higher powers of y can be replaced by polynomials in x, and h_D has the desired shape.

(ii) It is clear that H_D is a polynomial in $\mathbb{F}_q[x]$. By definition, the zeros of h_D are exactly $\varphi^j(P_1), \ldots, \varphi^j(P_r), j = 0, \ldots, n-1$, and its poles are $nr\mathcal{O}$. Therefore, $h_D \circ w = h_{D,1}(x) - y h_{D,2}(x)$ has $w(\varphi^j(P_1)), \ldots, w(\varphi^j(P_r)), j = 0, \ldots, n-1$ as zeros and $nr\mathcal{O}$ as poles. Since $H_D(x) = h_D(h_D \circ w)$ as functions on C, then H_D has precisely the zeros $\varphi^j(P_1), \ldots, \varphi^j(P_r), w(\varphi^j(P_1)), \ldots, w(\varphi^j(P_r))$ for $j = 0, \ldots, n-1$ and the poles $2nr\mathcal{O}$. But a function with this property is

$$b(x) = \prod_{i=1}^{r} \prod_{j=0}^{n-1} (x - X_i^{q^j}),$$

where the X_i are the x-coordinates of the P_i. Therefore, $H_D = b$ up to multiplication by a constant.

(iii) From the fact that $\deg H_D = nr$ and $\deg f = 2g+1$, we immediately deduce the bounds on the degrees. Now if r or n is even, then $\lfloor \frac{nr}{2} \rfloor = \frac{nr}{2}$ and $\lfloor \frac{nr-2g-1}{2} \rfloor = \frac{nr}{2} - g - 1$. Therefore $\deg(h_{D,1}^2) \leq nr$ and $\deg(f h_{D,2}^2) \leq nr-1$ and we see that $\deg h_{D,1} = \frac{nr}{2}$. An analogous computation for r and n both odd shows that in this case $\deg h_{D,2} = \frac{nr-1}{2} - g = \left\lfloor \frac{nr-2g-1}{2} \right\rfloor$.

(iv) If $r < g$, then the total number of coefficients required to store both $h_{D,1}$ and $h_{D,2}$ is

$$\deg h_{D_1} + \deg h_{D,2} + 2 \leq \left\lfloor \frac{nr}{2} \right\rfloor + \left\lfloor \frac{nr-2g-1}{2} \right\rfloor + 2 = nr - g + 1 \leq (n-1)(g-1).$$

If $r = g$ we normalize h_D in a suitable way, depending on the parity of g and n. Set

$$d_1 = \left\lfloor \frac{ng}{2} \right\rfloor, \quad d_2 = \left\lfloor \frac{ng-2g-1}{2} \right\rfloor.$$

If $n = 2$, then $\deg h_{D,2} \leq d_2 < 0$, hence $h_D = h_{D,1} = u$, where $[u,v]$ is the Mumford representation of D. Since $\deg u = d_1 = g$ and u is monic, we need g elements of $\mathbb{F}_q$ to represent it.

If n is odd and g is even, then we multiply h_D by a constant such that $h_{D,1}$ is monic. Then $h_{D,1}$ is given by $d_1 = \frac{ng}{2}$ elements of $\mathbb{F}_q$, namely the coefficients of $1, x, \ldots, x^{d_1-1}$, while $h_{D,2}$ is given by at most $d_2 + 1 = \frac{ng}{2} - g$ coefficients. Thus we need a total of $d_1 + d_2 + 1 = (n-1)g$ coefficients in $\mathbb{F}_q$ in order to store $h_{D,1}$ and $h_{D,2}$.

In the case that g and n are both odd, we multiply h_D by a constant such that $h_{D,2}$ is monic. Then $h_{D,1}$ is given by at most $d_1 + 1 = \frac{ng+1}{2}$ coefficients, and $h_{D,2}$ is given by the $d_2 = \frac{ng-1}{2} - g$ coefficients of $1, x, \ldots, x^{d_2-1}$. Again we need a total of $d_1 + d_2 + 1 = (n-1)g$ coefficients in $\mathbb{F}_q$ in order to store $h_{D,1}$ and $h_{D,2}$.

Let $\delta \in \mathbb{F}_2$ be 0 if $r < g$ and 1 if $r = g$. Then the polynomial h_D can be represented via $(n-1)g$ coefficients in $\mathbb{F}_q$, together with $\delta \in \mathbb{F}_2$.

(v) Let $F \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ be a reduced divisor such that $h_F = h_D \in \mathbb{F}_q(C)$. Then

$$\mathrm{Tr}(F) = \mathrm{div}(h_F) = \mathrm{div}(h_D) = \mathrm{Tr}(D) \in \mathrm{Div}_C(\mathbb{F}_q).$$

Write $\mathrm{Tr}(D) = \mathrm{Tr}(D_1) + \ldots + \mathrm{Tr}(D_t) = \mathrm{Tr}(F)$, where $\mathrm{Tr}(D_i) \in \mathrm{Div}_C(\mathbb{F}_q)$ are prime divisors by Lemma 2.16 (ii). By Lemma 2.16 (i), $\mathrm{Tr}^{-1}(\mathrm{Tr}(D_i)) = \{D_i, \varphi(D_i), \ldots, \varphi^{n-1}(D_i)\}$ for all i, hence $F = \varphi^{j_1}(D_1) + \ldots + \varphi^{j_t}(D_t)$ for some $j_1, \ldots, j_t \in \{0, \ldots, n-1\}$. The number of such F is $n^t \leq n^g$.

(vi) We have $h_{D,2}(x) \equiv 0 \bmod u_i$ if and only if $h_D(x,y) \equiv h_{D,1}(x) \equiv h_{w(D)}(x,y) \bmod u_i$. Since $D_i \leq \mathrm{Tr}(D)$, this is also equivalent to $w(D_i) \leq \mathrm{Tr}(D)$. Since D_i is prime, $w(D_i)$ is also prime and $w(D_i) \leq \mathrm{Tr}(D)$ if and only if $w(D_i) = \varphi^j(D_k)$ for some $j \in \{0, \ldots, n-1\}$ and some $k \in \{1, \ldots, t\}$ by Lemma 2.16 (i).

Now suppose that $n \neq 2$ and $w(D_i) = \varphi^j(D_i)$ for some $j \neq 0$. Then $u_i \in \mathbb{F}_q[x]$ and $-v_i = v_i^{\varphi^j}$, hence $-\nu = \nu^{\varphi^j}$ for all coefficients ν of v_i. But this implies $\nu^2 = (\nu^2)^{\varphi^j}$ and hence $\nu \in \mathbb{F}_{q^{2j}} \cap \mathbb{F}_{q^n} = \mathbb{F}_q$. Therefore we have $v_i \in \mathbb{F}_q[x]$, but this implies $v_i = 0$ and therefore $D_i = w(D_i)$.

(vii) Let $\mathrm{Tr}(D) = m\,\mathrm{Tr}(D_i) + \ell\,\mathrm{Tr}(w(D_i)) + \mathrm{Tr}(G)$ for some G, with $\mathrm{Tr}(D_i), \mathrm{Tr}(w(D_i)) \not\leq \mathrm{Tr}(G)$, and assume that $m \geq \ell$. The other case follows by symmetry. Then

$$\begin{aligned}
\mathrm{div}(N(u_i)^\ell h_{(m-\ell)D_i+G}) &= \ell\,\mathrm{Tr}(D_i) + \ell\,\mathrm{Tr}(w(D_i)) + (m-\ell)\,\mathrm{Tr}(D_i) + \mathrm{Tr}(G) \\
&= \ell\,\mathrm{Tr}(w(D_i)) + m\,\mathrm{Tr}(D_i) + \mathrm{Tr}(G) \\
&= \mathrm{Tr}(D) = \mathrm{div}(h_D),
\end{aligned}$$

so $h_D = N(u_i)^\ell h_{(m-\ell)D_i+G}$ up to multiplication by a constant. Hence $N(u_i)^\ell$ divides h_D. Notice that $(m-\ell)D_i+G$ has trace zero, since D does. Now assume that $N(u_i)$ also divides $h_{(m-\ell)D_i+G}$. Then $\operatorname{Tr}(D_i)+\operatorname{Tr}(w(D_i)) \leq (m-\ell)\operatorname{Tr}(D_i)+\operatorname{Tr}(G)$. Since $\operatorname{Tr}(w(D_i)) \not\leq \operatorname{Tr}(G)$ by assumption, this implies $\operatorname{Tr}(w(D_i)) \leq (m-\ell)\operatorname{Tr}(D_i)$ and therefore $\operatorname{Tr}(D_i) = w(\operatorname{Tr}(D_i))$. But this implies $D_i = w(D_i)$ by (vi), which contradicts our assumption. Therefore, $N(u_i)^\ell$ exactly divides h_D.

Conversely, assume that $h_D = N(u_i)^\ell h$ for some ℓ, where h is a polynomial and $N(u_i) \nmid h$. Then $\operatorname{Tr}(D) = \operatorname{div}(h_D) = \ell\operatorname{Tr}(D_i) + \ell\operatorname{Tr}(w(D_i)) + \operatorname{div}(h)$, and not both $\operatorname{Tr}(D_i)$ and $\operatorname{Tr}(w(D_i))$ are $\leq \operatorname{div}(h)$. Say $\operatorname{Tr}(w(D_i)) \not\leq \operatorname{div}(h)$, and k is maximal such that $k\operatorname{Tr}(D_i) \leq \operatorname{div}(h)$. Then

$$\operatorname{Tr}(D) = \ell\operatorname{Tr}(D_i) + \ell\operatorname{Tr}(w(D_i)) + k\operatorname{Tr}(D_i) + \operatorname{Tr}(G) = m\operatorname{Tr}(D_i) + \ell\operatorname{Tr}(w(D_i)) + \operatorname{Tr}(G)$$

where $m = \ell + k$ and $\operatorname{Tr}(D_i), \operatorname{Tr}(w(D_i)) \not\leq \operatorname{div}(h) - k\operatorname{Tr}(D_i) = \operatorname{Tr}(G)$. □

Remark 5.4. The results of Theorem 5.3 may be generalized to elliptic and hyperelliptic curves over fields of characteristic 2 by defining $H_D = h_D(h_D \circ w)$. It is not hard to check that we obtain a function h_D with the same properties as in (i)–(vii). Special caution needs to be used in adapting (vi). Since the existence of this function is the basis for the representation we propose in this chapter, all our results and methods work, with the necessary adjustments, over a finite field of any characteristic.

Let $[D] \in T_n \setminus \{0\}$. One of the consequences of Theorem 5.3 is that, depending on the parity of r and n, we know the exact degree of either $h_{D,1}$ or $h_{D,2}$. By making the appropriate one monic, we can represent h_D by $(n-1)g$ elements of $\mathbb{F}_q$, namely the coefficients of $h_{D,1}$ and $h_{D,2}$, plus one bit which indicates whether $r = g$. This is an optimal representation, since $(n-1)g\log_2 q + 1 \approx (n-1)g\log_2 q \approx \log_2 |T_n|$. See Remark 5.12 for an alternative approach that does not require storing the extra bit δ.

Corollary 5.5. *Under the assumptions and following the notation of Theorem 5.3, we represent a reduced divisor $D \in \operatorname{Div}^0_C(\mathbb{F}_{q^n})$ such that $[D] \in T_n$ as follows. Let $[u,v]$ be the Mumford representation of D, where $u(x) = u_g x^g + \ldots + u_1 x + u_0$ has degree $r \leq g$ (i.e. $u_g = \ldots = u_{r+1} = 0$) and $u_r = 1$. Set $d_1 = \lfloor \frac{ng}{2} \rfloor$ and $d_2 = \left\lfloor \frac{(n-2)g-1}{2} \right\rfloor$. Let $h_{D,1}(x) = \gamma_{d_1}x^{d_1} + \ldots + \gamma_1 x + \gamma_0$, $h_{D,2}(x) = \beta_{d_2}x^{d_2} + \beta_{d_2-1}x^{d_2-1} + \ldots + \beta_1 x + \beta_0$, where $h_{D,1}$ is monic if nr is even and $h_{D,2}$ is monic if nr is odd. If $r = g$ let $\delta = 1$, else let $\delta = 0$. Define:*

- *If $n = 2$, then*
$$\begin{array}{rcl} \mathcal{R}: T_2 & \longrightarrow & \mathbb{F}_q^g \times \mathbb{F}_2 \\ {[D]} & \longmapsto & (u_0, \ldots, u_{g-1}, \delta). \end{array}$$
- *If n is odd and g is even, then*
$$\begin{array}{rcl} \mathcal{R}: T_n & \longrightarrow & \mathbb{F}_q^{(n-1)g} \times \mathbb{F}_2 \\ {[D]} & \longmapsto & (\beta_0, \ldots, \beta_{d_2}, \gamma_0, \ldots, \gamma_{d_1-1}, \delta). \end{array}$$
- *If n and g are odd, then*
$$\begin{array}{rcl} \mathcal{R}: T_n & \longrightarrow & \mathbb{F}_q^{(n-1)g} \times \mathbb{F}_2 \\ {[D]} & \longmapsto & (\gamma_0, \ldots, \gamma_{d_1}, \beta_0, \ldots, \beta_{d_2-1}, \delta). \end{array}$$

Then $\mathcal{R}$ is an optimal representation for the non-zero elements of T_n. This representation identifies at most n^g elements, as stated in Theorem 5.3 (v).

Proof. If $n = 2$ the statement follows from Proposition 5.1. For $n \geq 3$, the statement is a direct consequence of the proof of Theorem 5.3 (iii) and (iv). Observe that if $r < g$, then

$$\deg h_{D,1} = \left\lfloor \frac{rn}{2} \right\rfloor \leq \frac{n(g-1)-1}{2} = \frac{ng}{2} - \frac{n+1}{2} \leq d_1 - 2$$

if g is even and

$$\deg h_{D,2} = \left\lfloor \frac{nr-2g-1}{2} \right\rfloor \leq \frac{n(g-1)-2g-2}{2} = \frac{(n-2)g-1}{2} - \frac{n+1}{2} \leq d_2 - 2$$

if g is odd. In particular, the polynomials $h_{D,1}$ and $h_{D,2}$ can always be represented using the number of coefficients claimed. Moreover, if $r < g$ some of the coefficients that we store are zero. In particular, $\gamma_i = 0$ for $i > \lfloor \frac{rn}{2} \rfloor$ and $\beta_i = 0$ for $i > \lfloor \frac{nr-2g-1}{2} \rfloor$. Since

$$d_1 + d_2 + 1 = \left\lfloor \frac{ng}{2} \right\rfloor + \left\lfloor \frac{(n-2)g-1}{2} \right\rfloor + 1 = (n-1)g,$$

we store the correct number of $\mathbb{F}_q$-coefficients in all cases.

Finally, the representation identifies at most n^g elements by Theorem 5.3 (v). □

Remarks 5.6. (i) Let $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ be a reduced divisor, $D = D_1 + \ldots + D_t$ with $D_i \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ reduced prime divisors. Notice that not all the divisors F of the form $F = \varphi^{j_1}(D_1) + \ldots + \varphi^{j_t}(D_t)$ for some $j_1, \ldots, j_t \in \{0, \ldots, n-1\}$ are reduced. E.g., let C be a hyperelliptic curve of genus 2 and let $P \in C(\mathbb{F}_{q^n}) \setminus C(\mathbb{F}_q)$ be a point. Then $\varphi(P) \neq P$ and $D = P + w(\varphi(P)) - 2\mathcal{O}$ is a reduced divisor. But a divisor $F = \varphi^{j_1}(P) + w(\varphi^{j_2}(P))$ is reduced if and only if $j_1 \neq j_2$.

Because of this, when decompressing $\mathcal{R}([D])$ one needs to discard all the divisor classes $[F] \in T_n$ where $\mathrm{Tr}(F) = \mathrm{Tr}(D)$ but F is not a reduced divisor. E.g., in the previous example we would discard the elements $F = \varphi^j(P) + w(\varphi^j(P)) \sim 0$. In Algorithm 5.3, for a given $\alpha = \mathcal{R}([D])$ we recover one reduced $F \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ such that $\mathcal{R}([F]) = \alpha$. Because of what we just discussed, F identifies the equivalence class of D.

(ii) If $r < g$, the choice of making the polynomial $h_{D,1}$ monic if nr is even and making $h_{D,2}$ monic otherwise is not necessary in order to have an optimal representation for the elements of T_n. However, this normalization makes the representation of the polynomial h_D standard, since otherwise h_D is only determined up to a non-zero constant multiple. In particular, two reduced divisors $D, F \in \mathrm{Div}_C$ such that $[D], [F] \in T_n$ have $\mathrm{Tr}(D) = \mathrm{Tr}(F) \in \mathrm{Div}_C$ if and only if $\mathcal{R}(D) = \mathcal{R}(F)$.

Remark 5.7. If only the u-polynomial of the Mumford representation of $[D]$ is available (e.g. when using the representation for elements of $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ which we discuss in Example 2.24), one can still compute $H_D = u^{1+\varphi+\ldots+\varphi^{n-1}} \in \mathbb{F}_q[x]$. From $H_D = h_{D,1}^2 - f h_{D,2}^2$, the polynomials $h_{D,1}, h_{D,2}$ may be recovered up to some indeterminacy by solving a linear system in their (unknown) coefficients. If $u = \prod u_i$ where u_i are irreducible over $\mathbb{F}_{q^n}$, then the indeterminacy comes from the fact that for each i we cannot distinguish between u_i and its Frobenius conjugates (since we only know the norm of u_i) and we do not know the sign of v_i. Hence representing a $[D] \in T_n$ via a pair of polynomials $h_{D,1}, h_{D,2}$ obtained as above still has optimal size, but it identifies up to $(2n)^g$ elements. In practice however, in order to compress it suffices to find one solution of the linear system.

5.2 Computing the rational function

It is easy to compute h_D using Cantor's Algorithm (see [Can87]) and a generalization of Miller's Algorithm (see [Mil04]) as follows. For $[D_1], [D_2] \in \mathrm{Pic}^0_C$ given in Mumford representation, Cantor's Algorithm returns a reduced divisor $D_1 \oplus D_2$ and a function a such that $D_1 + D_2 = D_1 \oplus D_2 + \mathrm{div}(a)$. We denote this as $\mathrm{Cantor}(D_1, D_2) = (D_1 \oplus D_2, a)$. Cantor's Algorithm is detailed in Algorithm 2.1. Lines 1-3 are the composition of the divisors to be added, and the result of this is reduced in lines 4-8.

The following iterative definition will allow us to compute h_D with a Miller-style algorithm. For a function h we denote by h^φ the application of the Frobenius field automorphism $\varphi : \overline{\mathbb{F}}_q \to \overline{\mathbb{F}}_q$ coefficient-wise to the function h.

Lemma 5.8. *Let $D = [u, v]$ be a divisor on C, and let $D_i = \varphi^i(D)$ for $i \geq 0$. Let $h^{(1)} = u$ as a function on C, and define recursively the functions*

$$h^{(i+j)} = h^{(i)} \cdot (h^{(j)})^{\varphi^i} \cdot a^{-1},$$

where a is given by Cantor's algorithm according to

$$w(D_0 \oplus \ldots \oplus D_{i-1}) + w(D_i \oplus \ldots \oplus D_{i+j-1}) = w(D_0 \oplus \ldots \oplus D_{i+j-1}) + \mathrm{div}(a)$$

for $i, j \geq 1$. Then for all $i \geq 1$ we have

$$\mathrm{div}(h^{(i)}) = D_0 + \ldots + D_{i-1} + w(D_0 \oplus \ldots \oplus D_{i-1}).$$

If $[D] \in T_n$, then

$$h^{(n-1)} = h_D.$$

Proof. It is clear that $\mathrm{div}(h^{(1)}) = \mathrm{div}(u) = D + w(D) = D_0 + w(D_0)$. By induction, we have

$$\begin{aligned}
\mathrm{div}(h^{(i+j)}) &= \mathrm{div}(h^{(i)}) + \mathrm{div}((h^{(j)})^{\varphi^i}) + \mathrm{div}(a^{-1}) \\
&= D_0 + \ldots + D_{i-1} + w(D_0 \oplus \ldots \oplus D_{i-1}) \\
&\quad + \varphi^i(D_0 + \ldots + D_{j-1} + w(D_0 \oplus \ldots \oplus D_{j-1})) \\
&\quad - w(D_0 \oplus \ldots \oplus D_{i-1}) - w(D_i \oplus \ldots \oplus D_{i+j-1}) + w(D_0 \oplus \ldots \oplus D_{i+j-1}) \\
&= D_0 + \ldots + D_{i+j-1} + w(D_0 \oplus \ldots \oplus D_{i+j-1}).
\end{aligned}$$

When $[D] \in T_n$, then $D_0 \oplus \ldots \oplus D_{n-1} = D \oplus \varphi(D) \oplus \ldots \oplus \varphi^{n-1}(D) = 0$, and therefore $w(D_0 \oplus \ldots \oplus D_{n-2}) = D_{n-1}$. Hence we have

$$\mathrm{div}\left(h^{(n-1)}\right) = D_0 + \ldots + D_{n-2} + w(D_0 \oplus \ldots \oplus D_{n-2}) = D_0 + \ldots + D_{n-1} = \mathrm{Tr}(D)$$

as claimed. □

The functions $h^{(i)}$ of Lemma 5.8 can be computed using a Miller-style algorithm, where at each step we apply Cantor's Algorithm to compute the function a. When $[D] \in T_n$, then the $(n-1)$-th iteration $h^{(n-1)}$ agrees with the desired function h_D. Hence h_D can easily be computed with a Miller-style algorithm, as in Algorithm 5.1.

Theorem 5.9. *Algorithm 5.1 computes h_D correctly. Computing h_D has a complexity of $O(g^4 \log_2 n + g^3 n + g^{\log_2 3} 3^{\log_2 n})$ operations in $\mathbb{F}_{q^n}$.*

Algorithm 5.1 Miller-style double and add algorithm for computing h_D

```
Input: $[D] = [u, v] \in T_n$ and $n - 1 = \sum_{j=0}^{s} n_j 2^j$
Output: $h_D$
1: $h \leftarrow u, R \leftarrow w(D), Q \leftarrow w(\varphi(D)), i \leftarrow 1$
2: for $j = s-1, s-2, \dots, 1, 0$ do
3:     $(R, a) \leftarrow \text{Cantor}(R, \varphi^i(R)), h \leftarrow h \cdot h^{\varphi^i} \cdot a^{-1}, Q \leftarrow \varphi^i(Q), i \leftarrow 2i$
4:     if $n_j = 1$ then
5:         $(R, a) \leftarrow \text{Cantor}(R, Q), h \leftarrow h \cdot u^{\varphi^i} \cdot a^{-1}, Q \leftarrow \varphi(Q), i \leftarrow i + 1$
6:     end if
7: end for
8: return $h$
```

Proof. Correctness: Denote by R_i, Q_i, h_i, a_i the values of the variables R, Q, h, a at step i of the execution of the algorithm. We claim that for every i we have $R_i = w(D_0 \oplus \ldots \oplus D_{i-1})$, $Q_i = w(D_i)$, and $h_i = h^{(i)}$. We prove the claim by induction on $i \geq 1$. For $i = 1$ there is nothing to prove, so we assume that the statement holds for i and prove it for $2i$ and $2i + 1$. In a "doubling" step, we compute

$$(R_{2i}, a_{2i}) = \text{Cantor}(R_i, \varphi^i(R_i)) = \text{Cantor}(w(D_0 \oplus \ldots \oplus D_{i-1}), w(D_i \oplus \ldots \oplus D_{2i-1})),$$

hence $R_{2i} = w(D_0 \oplus \ldots \oplus D_{2i-1})$. Moreover,

$$Q_{2i} = \varphi^i(Q_i) = \varphi^i(w(D_i)) = w(D_{2i}).$$

Finally, by Lemma 5.8

$$h_{2i} = h_i \cdot h_i^{\varphi^i} \cdot a_{2i}^{-1} = h^{(i)} \cdot (h^{(i)})^{\varphi^i} \cdot a_{2i}^{-1} = h^{(2i)}.$$

In an "addition" step, we compute

$$(R_{2i+1}, a_{2i+1}) = \text{Cantor}(R_{2i}, Q_{2i}) = \text{Cantor}(w(D_0 \oplus \ldots \oplus D_{2i-1}), w(D_{2i})),$$

hence $R_{2i+1} = w(D_0 \oplus \ldots \oplus D_{2i})$. Moreover,

$$Q_{2i+1} = \varphi(Q_{2i}) = \varphi(w(D_{2i})) = w(D_{2i+1}).$$

Finally,

$$h_{2i+1} = h_{2i} \cdot u^{\varphi^i} \cdot a_{2i+1}^{-1} = h^{(2i)} \cdot (h^{(1)})^{\varphi^{2i}} \cdot a_{2i+1}^{-1} = h^{(2i+1)}$$

according to Lemma 5.8. So the output of Algorithm 5.1 is $h^{(n-1)}$, a function with divisor $\text{Tr}(D)$, as desired.

Complexity: The algorithm takes $\log_2(n-1)$ iterations, which we approximate by $\log_2 n$. We concentrate only on the doublings, since they dominate the complexity of each step. The crucial operations are the execution of Cantor's Algorithm and the computation of $h^{(2i)}$ from $h^{(i)}$.

Let us first consider Cantor's Algorithm. According to [Can87], the algorithm has a complexity of $O(g^2 \log g)$ field operations. This does, however, not include the computation of the function a. Nevertheless, computing a according to the explicit formula given in equation (5.1) is not more expensive than computing h, since the computation of h involves multiplication by a^{-1}. For this reason, we disregard the computation of a in the complexity analysis. However, we study the shape of a, since this will be important for the analysis of the computation of h.

We assume that the input divisors to Cantor's Algorithm have u-polynomials of degree g, and that they are coprime (this is true generically). Then, at the end of the composition (after line 3 in Algorithm 2.1), we have $\deg u = 2g$ and $a = 1$. Let us call $[u_0, v_0]$ and $a_0 = 1$ the input to the reduction procedure (lines 4-8 of Algorithm 2.1) and u_i, v_i, a_i the values of u, v, a after the i-th iteration of the while loop. Then, following through the algorithm, one can easily check that

$$a_i = \begin{cases} u_0 \frac{(y-v_1)(y-v_3)\cdots(y-v_{i-2})}{(y+v_0)(y+v_2)\cdots(y+v_{i-1})} & \text{if } i \text{ is odd} \\ \frac{(y-v_0)(y-v_2)\cdots(y-v_{i-2})}{(y+v_1)(y+v_3)\cdots(y+v_{i-1})} & \text{if } i \text{ is even.} \end{cases} \tag{5.1}$$

Since in most cases the degree of u decreases by 2 at each step, as observed already by Cantor [Can87], and since we assume that u_0 has degree $2g$, we expect to go through about $g/2$ reduction steps. Therefore, the final a has about $g/4$ terms in both the numerator and the denominator. Since $\deg v_i \leq 2g$ for all i and computing modulo the curve equation, we get

$$a = \frac{b(x) + yc(x)}{d(x) + ye(x)}$$

where $b, c, d, e \in \mathbb{F}_{q^n}[x]$ are polynomials of degree in the order of g^2.

Next we analyze the computation of $h^{(2i)}$ as $h^{(i)} \cdot (h^{(i)})^{\varphi^i} \cdot a^{-1}$. Notice that $h^{(i)}$ is a polynomial for all i, since the corresponding principal divisor has its only pole at infinity. Therefore, by using the curve equation we obtain $h^{(i)} = h_1^{(i)} + yh_2^{(i)}$. By an inductive argument, it is easy to show that ig is a good approximation of (the upper bound on) the degrees of $h_1^{(i)}$ and $h_2^{(i)}$. Writing $h^{(i)} \cdot (h^{(i)})^{\varphi^i} \cdot (d + ye) = h_1' + yh_2'$ with $\deg h_1'$ and $\deg h_2'$ in the order of $g^2 + ig$, we obtain

$$(h_1^{(2i)} + yh_2^{(2i)})(b + yc) = h_1' + yh_2',$$

and hence

$$h_1^{(2i)} = \frac{bh_1' - fch_2'}{b^2 - fc^2}, \quad h_2^{(2i)} = \frac{h_2' - ch_1^{(2i)}}{b},$$

where the divisions are exact since the results must be polynomials. The most expensive multiplications involved are those by h_1' and h_2', since those have the largest degree, namely about $g^2 + ig$. Using Karatsuba multiplication, we can thus compute the numerators and denominators above in a total of $O((g^2 + ig)^{\log_2 3})$ operations. The two long divisions take $O(g^3(i + g))$ each, since both numerators have degree in the order of $g(i + g)$ and both denominators have degree in the order of g^2. Hence the entire computation of $h^{(2i)}$ takes $O(g^4 + g^3 i + (gi)^{\log_2 3})$ operations.

Finally, we sum over all $\log_2 n$ steps to obtain a total complexity of

$$O\left(\sum_{i=0}^{\log_2 n} (g^4 + g^3 2^i + (g2^i)^{\log_2 3})\right) = O\left(g^4 \log_2 n + g^3 n + g^{\log_2 3} 3^{\log_2 n}\right)$$

as claimed. □

Remark 5.10. It is also possible to determine h_D by solving a linear system of size about $gn \times gn$ for the coefficients. Using standard Gaussian elimination techniques, this has complexity $O((gn)^3)$. This is larger in n but smaller in g than the complexity of Algorithm 5.1. Therefore, the linear algebra method is preferable when n is small and g is large.

5.3 Compression and decompression algorithms

We propose the compression and decompression algorithms detailed in Algorithms 5.2 and 5.3. We denote by lc the leading coefficient of a polynomial. *We only discuss the case* $n \geq 3$, since in the case $n = 2$ the representation consists of $u(x)$ (see Proposition 5.1, Theorem 5.3 (iv), and Corollary 5.5).

The compression algorithm follows immediately from the results of the previous section. The strategy of the decompression algorithm is as follows. From the input $\alpha = \mathrm{Compress}(D)$, we recompute $h_{D,1}$ and $h_{D,2}$ and then H_D. Then we factor H_D in order to obtain the u-polynomials of (one Frobenius conjugate of each of) the $\mathbb{F}_{q^n}$-rational prime divisors in D. This is consistent with the fact that $\mathrm{Tr}(D)$ only contains information about the conjugacy classes of these prime divisors. Afterwards, we compute the corresponding v-polynomial for each u-polynomial. In this way, if $D = D_1 + \ldots + D_t$ is the decomposition of D as a sum of $\mathbb{F}_{q^n}$-rational prime divisors, we recover an $\mathbb{F}_{q^n}$-rational prime divisor $D'_1 + \ldots + D'_t$, where each D'_i is one of the Frobenius conjugates of D_i. The divisor $D'_1 + \ldots + D'_t$ corresponds to the class $\mathcal{R}^{-1}(\alpha)$, since every $F \in \mathcal{R}^{-1}(\alpha)$ is of the form $\varphi^{j_1}(D'_1) + \ldots + \varphi^{j_t}(D'_t)$ for some $j_1, \ldots, j_t \in \{0, \ldots, n-1\}$ (see Theorem 5.3 (v)). Since we know that α is the representation of a reduced divisor, we compute a reduced representative $D'_1 + \ldots + D'_t$ of the class $\mathcal{R}^{-1}(\alpha)$ (see also Remark 5.6).

In the compression and the decompression algorithms we treat the last element of the vector which gives the representation sometimes as an element of $\mathbb{F}_q$ and sometimes as an element of $\mathbb{F}_2$, since it is an element of $\mathbb{F}_q$ which is either 0 or 1.

Algorithm 5.2 Compression, $n \geq 3$

```
Input: $[D] = [u, v] \in T_n$
Output: Representation $(\alpha_0, \ldots, \alpha_{(n-1)g}) \in \mathbb{F}_q{}^{(n-1)g} \times \mathbb{F}_2$ of $[D]$
 1: $r \leftarrow \deg u$
 2: compute $h_D(x, y) = h_{D,1}(x) + y h_{D,2}(x)$ (see Algorithm 5.1)
 3: $d_1 \leftarrow \lfloor \frac{ng}{2} \rfloor$
 4: $d_2 \leftarrow \lfloor \frac{ng-2g-1}{2} \rfloor$
 5: if $r$ even then
 6:     $h_{D,1} \leftarrow h_{D,1} / \mathrm{lc}(h_{D,1})$    ▷ Notation: $h_{D,1} = \gamma_{d_1} x^{d_1} + \gamma_{d_1-1} x^{d_1-1} + \ldots + \gamma_0$ monic
 7:     $h_{D,2} \leftarrow h_{D,2} / \mathrm{lc}(h_{D,1})$    ▷ Notation: $h_{D,2} = \beta_{d_2} x^{d_2} + \beta_{d_2-1} x^{d_2-1} + \ldots + \beta_1 x + \beta_0$
 8: else
 9:     $h_{D,1} \leftarrow h_{D,1} / \mathrm{lc}(h_{D,2})$    ▷ Notation: $h_{D,1} = \gamma_{d_1} x^{d_1} + \gamma_{d_1-1} x^{d_1-1} + \ldots + \gamma_1 x + \gamma_0$
10:     $h_{D,2} \leftarrow h_{D,2} / \mathrm{lc}(h_{D,2})$    ▷ Notation: $h_{D,2} = \beta_{d_2} x^{d_2} + \beta_{d_2-1} x^{d_2-1} + \ldots + \beta_0$ monic
11: end if
12: if $g$ even then
13:     return $(\beta_0, \ldots, \beta_{d_2}, \gamma_0, \ldots, \gamma_{d_1})$
14: else
15:     return $(\gamma_0, \ldots, \gamma_{d_1}, \beta_0, \ldots, \beta_{d_2})$
16: end if
```

Theorem 5.11. (*i*) *Compression Algorithm 5.2 computes a unique optimal representation of a given* $[D] \in T_n$*, for* D *a reduced divisor. It takes* $O(g^4 \log_2 n + g^3 n + g^{\log_2 3} 3^{\log_2 n})$ *operations in* $\mathbb{F}_{q^n}$*.*

(*ii*) *Decompression Algorithm 5.3 operates correctly: for any input* Compress(D)*, where* $[D] \in T_n$*, it returns a reduced divisor* D' *such that* $[D'] \in T_n$ *and* Compress$(D) =$ Compress(D')*.*

(*iii*) *Decompression Algorithm 5.3 takes* $O((ng)^{1+\log_2 3} \log(q^n ng))$ *operations in* $\mathbb{F}_{q^n}$*.*

Algorithm 5.3 Decompression, $n \geq 3$

```
Input: $(\alpha_0, \ldots, \alpha_{(n-1)g}) \in {\mathbb{F}_q}^{(n-1)g} \times \mathbb{F}_2$
Output: one reduced $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ s.t. $[D] \in T_n$ has representation $(\alpha_0, \ldots, \alpha_{(n-1)g})$
 1: $d_1 \leftarrow \lfloor \frac{ng}{2} \rfloor$
 2: $d_2 \leftarrow \lfloor \frac{ng-2g-1}{2} \rfloor$
 3: if $g$ even then
 4:     $h_{D,1}(x) \leftarrow \alpha_{(n-1)g} x^{d_1} + \ldots + \alpha_{d_2+2} x + \alpha_{d_2+1}$
 5:     $h_{D,2}(x) \leftarrow \alpha_{d_2} x^{d_2} + \alpha_{d_2-1} x^{d_2-1} + \ldots + \alpha_1 x + \alpha_0$
 6: else
 7:     $h_{D,1}(x) \leftarrow \alpha_{d_1} x^{d_1} + \ldots + \alpha_1 x + \alpha_0$
 8:     $h_{D,2}(x) \leftarrow \alpha_{(n-1)g} x^{d_2} + \ldots + \alpha_{d_1+2} x + \alpha_{d_1+1}$
 9: end if
10: $H_D(x) \leftarrow h_{D,1}(x)^2 - f(x) h_{D,2}(x)^2$
11: factor $H_D(x) = U_1(x)^{e_1} \cdot \ldots \cdot U_m(x)^{e_m}$ with $U_i \in \mathbb{F}_q[x]$ irred., distinct, $e_i \in \{1, \ldots, gn\}$
12: $L \leftarrow$ empty list
13: for $i = 1, \ldots, m$ do
14:     if $U_i(x)$ is irreducible over $\mathbb{F}_{q^n}$ then      ▷ $U_i$ comes from $\mathbb{F}_q$-rational prime divisor
15:         $e_i \leftarrow e_i / n$
16:     end if
17:     $U(x) \leftarrow$ one irreducible factor over $\mathbb{F}_{q^n}$ of $U_i(x)$
18:     if $h_{D,2}(x) \not\equiv 0 \bmod U(x)$ then
19:         $V(x) \leftarrow -h_{D,1}(x) h_{D,2}(x)^{-1} \bmod U(x)$
20:         append $[U(x), V(x)]$ to $L, e_i$ times
21:     else                                          ▷ $h_{D,2}(x) \equiv 0 \bmod U(x)$
22:         if $f(x) \equiv 0 \bmod U(x)$ then            ▷ $V(x) = 0$ and $D_i = w(D_i)$
23:             append $[U(x), 0], [U(x)^{\varphi}, 0], \ldots, [U(x)^{\varphi^{e_i-1}}, 0]$ to $L$
24:         else                                      ▷ $V(x) \neq 0$ and $D_i \neq w(D_i)$
25:             compute $\ell, h'_D$ such that $h_D = U_i(x)^{\ell} h'_D$ and $U_i(x) \nmid h'_D$
26:             if $\ell < e_i/2$ then
27:                 $V(x) \leftarrow -h'_{D,1}(x) h'_{D,2}(x)^{-1} \bmod U(x)$
28:                 append $[U(x), V(x)]$ to $L, e_i - \ell$ times
29:                 append $[U(x)^{\varphi}, -V(x)^{\varphi}]$ to $L, \ell$ times
30:             else                                  ▷ $\ell = e_i/2$
31:                 $V(x) \leftarrow \sqrt{f(x)} \bmod U(x)$
32:                 append $[U(x), V(x)], [U(x)^{\varphi}, -V(x)^{\varphi}]$ to $L, \ell$ times
33:             end if
34:         end if
35:     end if
36: end for                                           ▷ Notation: $L = [D_1, \ldots, D_t]$
37: return $D = D_1 + \ldots + D_t$
```

Proof. (i) Correctness and optimality of the representation computed by Algorithm 5.2 follow from Corollary 5.5. The complexity of computing the representation is the same as the complexity of computing h_D, which is given in Theorem 5.9.

(ii) Let $D = D_1 + \ldots + D_t$, where D_i are reduced prime divisors defined over $\mathbb{F}_{q^n}$, possibly not all distinct. The reduced divisors D' such that $[D'] \in T_n$ and $\text{Compress}(D) = \text{Compress}(D')$ are exactly those for which $h_{D'} = h_D$. By Theorem 5.3 (v), they are of the form $D' = \varphi^{j_1}(D_1) + \ldots + \varphi^{j_t}(D_t)$ for some $0 \le j_1, \ldots, j_t \le n-1$, and we will show that Algorithm 5.3 computes such a divisor D' for some $j_1, \ldots, j_t$.

Let $[u_i, v_i]$ be the Mumford representation of D_i, $u_i \in \mathbb{F}_{q^n}[x]$ irreducible. We have

$$H_D(x) = H_{D'}(x) = \prod_{i=1}^{t} u_i^{1+\varphi+\ldots+\varphi^{n-1}} = \prod_{i=1}^{m} U_i(x)^{e_i},$$

where $U_i \in \mathbb{F}_q[x]$ are irreducible and $U_i \neq U_j$ if $i \neq j$, $m \le t$. Up to reindexing, $U_i = u_i$ if $u_i \in \mathbb{F}_q[x]$ and $U_i = N(u_i)$ otherwise, for $i \le m$. If $u_i \in \mathbb{F}_q[x]$, then $u_i^{1+\varphi+\ldots+\varphi^{n-1}} = u_i^n = U_i^n$, hence $n \mid e_i$ and we replace e_i by e_i/n, since $\text{Tr}(D_i) = nD_i$. Notice that by Lemma 2.16 (ii) U_i is an $\mathbb{F}_q[x]$-irreducible factor of $H_D(x)$ independently of whether $u_i \in \mathbb{F}_q[x]$ or not. Notice moreover that $u_i \in \mathbb{F}_q[x]$ if and only if U_i is irreducible in $\mathbb{F}_{q^n}[x]$. Conversely, U_i is reducible in $\mathbb{F}_{q^n}[x]$ if and only if $u_i \in \mathbb{F}_{q^n}[x]$ is one of its irreducible factors. Summarizing, each D_i corresponds exactly to a set of n $\mathbb{F}_{q^n}[x]$-irreducible factors of H_D, and these factors can be correctly grouped by first computing the $\mathbb{F}_q[x]$-factorization of $H_D = N(u)$.

Fix $i \in \{1, \ldots, m\}$ and let $U(x)$ be an $\mathbb{F}_{q^n}[x]$-irreducible factor of $U_i(x)$, i.e., $U(x)$ is a Frobenius conjugate of $u_i(x)$. We now discuss: how to compute the corresponding $V(x)$, so that $[U, V]$ is the Mumford representation of some Frobenius conjugate of D_i, and how to determine the cardinality m of the set of $j \in \{1, \ldots, t\}$ such that $D_j \le D$ is a Frobenius conjugate of D_i, respectively the cardinality ℓ of the set of $j \in \{1, \ldots, t\}$ such that $D_j \le D$ is a Frobenius conjugate of $w(D_i)$. If $D_i \neq w(D_i)$ we have that $H_D = N(u)$ is divisible by $U_i^m U_i^\ell$ and by no higher power of U_i by Theorem 5.3 (ii), hence $e_i = m + \ell$. If $D_i = w(D_i)$ by definition $m = \ell$ and $H_D = N(u)$ is divisible by U_i^m and by no higher power of U_i by Theorem 5.3 (ii), hence $e_i = m$.

By Theorem 5.3 (vi), if $U \nmid h_{D,2}$ then $\ell = 0$ and no Frobenius conjugate of $w(D_i)$ appears among the prime divisors $D_j \le D$. Moreover, there exist polynomials $k(x), l(x) \in \mathbb{F}_{q^n}[x]$ such that $k(x)h_{D,2} = 1 + l(x)U(x)$. Hence $k(x)(h_{D,1}(x) + yh_{D,2}(x)) \equiv y + k(x)h_{D,1} \bmod U$. Since $h_{D,1} + yh_{D,2} \equiv 0 \bmod (U, y-V)$, then $y-V$ divides $y + k(x)h_{D,1} \bmod U$. It follows that $y - V \equiv y + k(x)h_{D,1} \bmod U$, hence

$$V \equiv -h_{D,1}h_{D,2}^{-1} \bmod U.$$

Notice that in this case $V \neq 0$, since $D_i \neq w(D_i)$, and $m = e_i$.

If $U \mid h_{D,2}$, it follows from Theorem 5.3 (vi) that $w(D_i) = \varphi^j(D_k)$ for some $0 \le j \le n-1$ and $1 \le k \le t$. In other words, both $[u_i, v_i] = D_i$ and some Frobenius conjugate of $[u_i, -v_i] = w(D_i) = \varphi^j(D_k)$ appear in D. We distinguish the cases

(a) $D_i = w(D_i)$,

(b) $D_i \neq w(D_i)$.

Case (a) is treated in lines 22–23 of the algorithm. In this case $m = \ell = e_i$. Since $D_i = w(D_i)$ is equivalent to $v_i = 0$, case (a) can be detected by checking whether $U \mid f$. If this is the case, it suffices to set $V = 0$.

Case (b) is treated in lines 25–33 of the algorithm. In this case $i \neq k$ by Theorem 5.3 (vi) since $n \neq 2$. By Theorem 5.3 (vii), $\mathrm{Tr}(D) = m\,\mathrm{Tr}(D_i) + \ell\,\mathrm{Tr}(w(D_i)) + \mathrm{Tr}(G)$ where $\mathrm{Tr}(D_i), \mathrm{Tr}(w(D_i)) \not\leq \mathrm{Tr}(G)$ and $s := \min\{m, \ell\}$ may be computed as the exponent for which $U_i^s \mid h_D$ and $U_i^{s+1} \nmid h_D$. Let $h_D = U_i^s h'_D$, $h'_D = h'_{D,1} + y h'_{D,2}$. Notice that $U, U_i \nmid h'_{D,2}$, since $\mathrm{Tr}(D_i) + \mathrm{Tr}(w(D_i)) \not\leq \mathrm{div}(h'_D)$. Then: $s = m = \ell = e_i/2$ if and only if $\mathrm{Tr}(D_i), \mathrm{Tr}(w(D_i)) \not\leq \mathrm{div}(h'_D)$ if and only if $U \nmid V^2 - f$, where $V = -h'_{D,1}{h'_{D,2}}^{-1} \bmod U$. In this case, we can let $V = \sqrt{f} \bmod U$, and D contains exactly $e_i/2$ Frobenius conjugates of $[U, V]$ and $e_i/2$ Frobenius conjugates of $[U, -V]$. If instead $s = \ell < m$, then $h_D = U_i^\ell h'_D$ and $\mathrm{div}(h'_D) \geq \mathrm{Tr}(D_i)$, $\mathrm{div}(h'_D) \not\geq \mathrm{Tr}(w(D_i))$. Therefore, $U \nmid h'_{D,2}$ and V can be computed as $V = -h'_{D,1}{h'_{D,2}}^{-1} \bmod U$. In this case, D contains exactly $m = e_i - \ell$ Frobenius conjugates of $[U, V]$ and ℓ Frobenius conjugates of $[U, -V]$.

Notice that $U \nmid h_{D,2}$ is the generic case, and it is treated in lines 19–20 of the algorithm. Lines 21–34 are only needed to treat the special case $U \mid h_{D,2}$ and distinguish the special cases that we just discussed.

Finally, we show that the D' returned by Algorithm 5.3 is reduced. To this end, we check that the algorithm does not add both a divisor and its involution to the list L, and in particular when a divisor is 2-torsion, we check that it is added with multiplicity 1. Since for each i such that $U \nmid h_{D,2}$ we have computed a unique $V \neq 0$, we only need to consider the cases where $U \mid h_{D,2}$. In case (a) we have $D_i = w(D_i)$, and we need to check that $D'_i, \varphi(D'_i), \ldots, \varphi^{e_i-1}(D'_i)$ are distinct, where $D'_i = [U, 0]$. In particular, we check that $e_i < n$. But if D'_i, hence D_i, were $\mathbb{F}_q$-rational or $e_i > n$, then D would not be reduced. In case (b) we have $D_i \neq w(D_i)$ and hence $w(\varphi(D_i)) \neq \varphi(D_i)$, so we may add several times $D'_i = [U, V]$ and $w(\varphi(D'_i)) = [U^\varphi, -V^\varphi]$. Furthermore, we have $D'_i \neq \varphi(D'_i)$. Indeed, if $D'_i = \varphi(D'_i)$ then it follows that D'_i, D_i and D_k are $\mathbb{F}_q$-rational. Hence $D_i = w(D_k)$, a contradiction to the fact that D is reduced, since $i \neq k$ in this case.

(iii) We assume that the degrees of $h_{D,1}$ and $h_{D,2}$ are maximal, which is the generic case. The complexity of the algorithm is dominated by the polynomial factorizations. The factorization of H_D, which has degree ng, takes $O((ng)^{1+\log_2 3}\log(qng))$ operations over $\mathbb{F}_q$ (see [GvzG99, Theorem 14.14]). In the loop over i, a polynomial U_i must be factored over $\mathbb{F}_{q^n}$ in each iteration. Write $\deg U_i = k_i$. Factoring U_i has complexity $O((k_i)^{1+\log_2 3}\log(q^n k_i))$ over $\mathbb{F}_{q^n}$ (as above). Inverting $h_{D,2}$ modulo U is in $O(k_i^2)$ and is therefore cheaper. Hence the overall complexity of the loop is

$$O\left(\sum_{i=1}^{\ell} k_i^{1+\log_2 3}\log(q^n k_i)\right).$$

This is largest in the extreme case where $\ell = 1$ and $k_1 = ng$, which yields the statement of the theorem. □

Remark 5.12. The representation computed by Algorithm 5.2 has size $(n-1)g\log_2 q + 1$. If one chooses to work only with divisors of the form $D = P_1 + \ldots + P_g - g\mathcal{O}$, then the last bit may be dropped and we have a representation of size $(n-1)g\log_2 q$. Divisor classes whose reduced representative has this form constitute the majority of the elements of T_n. Moreover, there are cases in which the trace zero subgroup consists only of divisor classes represented by reduced divisors of this shape. This is the case for elliptic curves, where $r = 1$ if $D \neq 0$. Moreover, Lange [Lan04b, Theorem 2.2] proves that for $g = 2$ and $n = 3$, all non-trivial elements of T_3 are represented by reduced divisors with $r = 2 = g$.

Remark 5.13. On the basis of the results discussed in Chapter 4.4, we suggest the following procedure for adding points in compressed coordinates: Decompress the point, perform the operation in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$, and compress the result. Since our compression and decompression algorithms are very efficient, this adds only little overhead. Moreover, scalar multiplication is considerably more efficient for trace zero points than for general points in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$, due to the speed-up using the Frobenius endomorphism.

5.4 Explicit equations for $g = 2, n = 3$

For small n and g, it is possible to give explicit equations for compression and decompression. In addition to making the computation more efficient, they allow us to perform precise operation counts and thus to compare our method to the other existing compression methods in Section 5.5. We postpone explicit equations for elliptic curves to Chapter 6 and consider here only the case of trace zero subgroups associated to hyperelliptic curves of genus 2 with respect to field extensions of degree 3.

For these parameters, the trace zero variety was studied in detail by Lange [Lan01, Lan04b]. One of her results is that divisor classes in T_3 are always represented by reduced divisors of a certain shape.

Theorem 5.14 ([Lan04b, Theorem 2.2])**.** *Assume that C has genus 2 and that $2, 3 \nmid |\mathrm{Pic}^0_C(\mathbb{F}_{q^3})|$. Then all non-trivial elements of T_3 are represented by reduced divisors of the form*

$$P_1 + P_2 - 2\mathcal{O} \notin \mathrm{Div}_C(\mathbb{F}_q),$$

where $P_1, P_2 \neq \mathcal{O}$ and $P_1 \notin \{P_2, \varphi(P_2), \varphi^2(P_2)\}$.

Corollary 5.15. *Assume that C has genus 2 and that $2, 3 \nmid |\mathrm{Pic}^0_C(\mathbb{F}_{q^3})|$. Then all non-trivial elements of T_3 are represented by reduced divisors of the form $D = P_1 + P_2 - 2\mathcal{O} \notin \mathrm{Div}_C(\mathbb{F}_q)$, and one of the following mutually exclusive facts holds:*

(i) $P_1, P_2 \in C(\mathbb{F}_{q^3}) \setminus \{\mathcal{O}\}$ *and* $P_1 \in \{w(\varphi(P_2)), w(\varphi^2(P_2))\}$,

(ii) $P_1, P_2 \in C(\mathbb{F}_{q^3}) \setminus \{\mathcal{O}\}$ *and* $P_1 \notin \{P_2, \varphi(P_2), \varphi^2(P_2), w(\varphi(P_2)), w(\varphi^2(P_2))\}$,

(iii) $P_1 \in C(\mathbb{F}_{q^6}) \setminus C(\mathbb{F}_{q^3})$ *and* $P_2 = \varphi^3(P_1)$.

Let $[u, v]$ be the Mumford representation of $[D]$. Then in cases (ii) and (iii) the divisor $D + \varphi(D)$ is semi-reduced and $u \nmid h_{D,2}$, in particular $h_{D,2} \neq 0$.

Proof. (ii) By contradiction, assume that $h_{D,2} \equiv 0 \bmod u$. Let $P_j = (X_j, Y_j)$, $j = 1, 2$. $P_j - \mathcal{O} \in \mathrm{Div}_C(\mathbb{F}_{q^3})$ is a reduced prime divisor. Since $h_{D,2}(X_j) = 0$, by Theorem 5.3 (vi) we have $w(P_j) = \varphi^i(P_j)$. Then $X_j \in \mathbb{F}_{q^3} \cap \mathbb{F}_{q^i} = \mathbb{F}_q$ and $Y_j \in \mathbb{F}_{q^3} \cap \mathbb{F}_{q^{2i}} = \mathbb{F}_q$. Hence $D = P_1 + P_2 - 2\mathcal{O} \in \mathrm{Div}_C(\mathbb{F}_q)$, which contradicts Theorem 5.14.

(iii) Let $D = P_1 + P_2 - 2\mathcal{O} \in \mathrm{Div}_C(\mathbb{F}_{q^3}) \setminus \mathrm{Div}_C(\mathbb{F}_q)$, and assume that $P_1 \notin C(\mathbb{F}_{q^3})$. Then also $P_2 \notin C(\mathbb{F}_{q^3})$ and $\varphi^3(D) = D$ forces $P_1 = \varphi^3(P_2)$ and $P_2 = \varphi^3(P_1)$. If $w(D) = \varphi^i(D)$ for some $i = 1, 2$, then either $w(P_1) = \varphi^i(P_1)$ or $w(P_1) = \varphi^{i+3}(P_1)$. Hence $P_1 = (X, Y)$ and $\varphi^j(P_1) = (X^{q^j}, Y^{q^j})$ lie on the same vertical line for some $j \in \{i, i+3\}$, therefore $X = X^{q^j} \in \mathbb{F}_{q^6} \cap \mathbb{F}_{q^j} \subseteq \mathbb{F}_{q^2}$ and $Y = -Y^{q^j} \in \mathbb{F}_{q^6} \cap \mathbb{F}_{q^{2j}} \subseteq \mathbb{F}_{q^2}$. This shows that $D \in \mathrm{Div}_C(\mathbb{F}_{q^2}) \cap \mathrm{Div}_C(\mathbb{F}_{q^3}) = \mathrm{Div}_C(\mathbb{F}_q)$, which contradicts Theorem 5.14. Therefore $w(D) \neq \varphi^i(D)$ for $i = 0, 1, 2$, which by Theorem 5.3 (vi) implies that $u \nmid h_{D,2}$, where $[u, v]$ is the Mumford representation of $[D]$. In particular, $h_{D,2} \neq 0$. □

We assume $2, 3 \nmid |\mathrm{Pic}^0_C(\mathbb{F}_{q^3})|$ throughout this section. Hence we know that the Mumford representation of all non-trivial elements of T_3 has a u-polynomial of degree 2 in $\mathbb{F}_{q^3}[x]$.

Furthermore, we assume that the characteristic of $\mathbb{F}_q$ *is not equal to 2 or 5.* In such a case, a simple transformation yields a curve equation of the shape

$$C : y^2 = x^5 + f_3x^3 + f_2x^2 + f_1x + f_0.$$

We assume here that C is given in this form, which slightly simplifies the explicit equations given below. Formulas for the more general case can be worked out easily.

Compression. We consider elements $0 \neq [D] = [u, v] \in T_3$ with $D = P_1 + P_2 - 2\mathcal{O}$ where u and u^φ are coprime. This is true under the additional assumption that the x-coordinates of both P_1, P_2 are not in $\mathbb{F}_q$. In addition we assume that $P_1 \notin \{w(\varphi(P_2)), w(\varphi^2(P_2))\}$, which by Corollary 5.15 implies that $h_{D,2}$ is not the zero polynomial. The other special cases can be worked out separately, and we do not treat them here.

Proposition 5.16. *Let $0 \neq [D] = [u, v] \in T_3$ such that $\gcd(u, u^\varphi) = 1$ and $h_{D,2}$ is not the zero polynomial. Let $[U, V]$ be the Mumford representation of the semi-reduced divisor $D + \varphi(D)$. Then*

$$h_D = y - V \text{ where } V = su + v,\ s \equiv (v^\varphi - v)/u \bmod u^\varphi.$$

Proof. The divisor $D+\varphi(D)$ is semi-reduced by Corollary 5.15. According to Theorem 5.3 (iii), we have $h_D = h_{D,1} + yh_{D,2}$ with $\deg h_{D,1} = 3$ and $\deg h_{D,2} \leq 0$. Since $h_{D,2} \neq 0$ by assumption, and after multiplication by a constant, we have $h_D = y-\gamma(x)$ where $\gamma \in \mathbb{F}_q[x]$ of degree 3. Now if $D = P_1 + P_2 - 2\mathcal{O}$ with $P_i = (X_i, Y_i)$, then $h_D(X_i^{q^j}, Y_i^{q^j}) = 0$ and hence $\gamma(X_i^{q^j}) = Y_i^{q^j}$ for $i = 1, 2,\ j = 0, 1, 2$. But V is the unique polynomial of degree ≤ 3 with $V(X_i^{q^j}) = Y_i^{q^j}$ for $i = 1, 2, j = 0, 1, 2$, and therefore $\gamma = V$.

In order to compute V, observe that V is the unique polynomial of degree $< \deg(uu^\varphi) = 4$ such that

$$V \equiv v \bmod u \quad \text{and} \quad V \equiv v^\varphi \bmod u^\varphi.$$

Keeping in mind that u, u^φ are coprime, and using the Chinese Remainder Theorem (or following the explicit formulas in [Lan05]), we get

$$V = su + v \quad \text{where} \quad s \equiv (v^\varphi - v)/u \bmod u^\varphi$$

with $\deg s \leq 1$ and hence $\deg V \leq 3$. □

Denoting $u(x) = x^2 + u_1x + u_0$ and $v(x) = v_1x + v_0$, we compute the compression $(\beta_0, \gamma_0, \gamma_1, \gamma_2, 1)$ of D according to the following formulas. We abbreviate

$$U_0 = u_0 - u_0^q,\ \ U_1 = u_1 - u_1^q,\ \ V_0 = v_0 - v_0^q,\ \ V_1 = v_1 - v_1^q.$$

Then

$$\begin{aligned}
d &= (U_1V_0 - U_0V_1)^{-1}\\
\beta_0 &= ((u_0u_1^q - u_0^qu_1)U_1 - U_0^2)d\\
\gamma_0 &= ((u_0v_0^q - u_0^qv_0)U_0 + (u_0^qu_1v_0 - u_0u_1^qv_0^q - u_0^{q+1}V_1)U_1)d\\
\gamma_1 &= ((u_0v_1^q - u_0^qv_1)U_0 + (u_1^qv_0 + u_0^qv_1^q)u_1U_1 + (u_0^qu_1 - u_0u_1^q)V_0\\
&\quad +(u_0v_1 + u_1v_0^q)(u_1^{2q} - u_1^{q+1}))d\\
\gamma_2 &= (((u_1 + u_1^q)U_1 - U_0)V_0 - (u_0u_1 - u_0^qu_1^q)V_1)d.
\end{aligned}$$

Counting squarings (S), multiplications (M), and inversions (I) but not the multiplication by constants (and therefore also not the application of the Frobenius), we see that computing these values in a straightforward way takes 2S+32M+1I in $\mathbb{F}_{q^3}$. This number could probably be optimized by regrouping the terms in a more sophisticated way. This is the total compression cost.

Decompression. Since decompression is dominated by factoring polynomials, we do not perform an exact operation count here. The algorithm computes

$$\begin{aligned}
S_1 &= -2\gamma_2 + \beta_0^2 \\
S_2 &= 2\gamma_1 + \gamma_2^2 \\
S_3 &= -2\gamma_0 - 2\gamma_1\gamma_2 + \beta_0^2 f_3 \\
S_4 &= 2\gamma_0\gamma_2 + \gamma_1^2 - \beta_0^2 f_2 \\
S_5 &= -2\gamma_0\gamma_1 + \beta_0^2 f_1 \\
S_6 &= \gamma_0^2 - \beta_0^2 f_0
\end{aligned}$$

over $\mathbb{F}_q$ to obtain $H_D = x^6 - S_1x^5 + S_2x^4 - S_3x^3 + S_4x^2 - S_5x + S_6$. In almost all cases we are decompressing a point of the shape that we have considered above for compression. H_D will either split over $\mathbb{F}_q$ into two factors of degree 3 or it will be irreducible over $\mathbb{F}_q$. Factoring H_D over $\mathbb{F}_q$ takes $O(\log q)$ operations in $\mathbb{F}_q$. Then we factor either two polynomials of degree 3 over $\mathbb{F}_{q^3}$ or one degree 6 polynomial over $\mathbb{F}_{q^3}$ in $O(\log q)$ operations in $\mathbb{F}_{q^3}$. In both cases, we then compute the corresponding v-polynomial(s). It follows that the overall complexity is $O(\log q)$ operations in $\mathbb{F}_q$.

5.5 Timings and comparison with other representations

Important achievements of this new representation are that it works for any prime n and any genus and can be made practical for large values of n and/or g. Moreover our decompression algorithm allows the unique recovery of one well-defined class of conjugates of the original point. For elliptic curves, such a class consists exactly of the Frobenius conjugates of the original point, and for higher genus curves, classes are as described in Theorem 5.3 (v) and Corollary 5.5. Identifying these conjugates is the natural choice from a mathematical point of view, since it respects the structure of our object and is compatible with scalar multiplication of points.

There are only three other known methods for point compression in trace zero varieties over elliptic curves, namely [Nau99], [Sil05], and the one from Chapter 4 of this thesis. While [Nau99] only applies to extension degree 3, [Sil05] and the one of Chapter 4 can be made practical for $n = 3, 5$. The approach of Chapter 4 allows unique recovery of an equivalence class for $n = 3$ and for most points for $n = 5$. The methods of [Nau99, Sil05] recover sets of points with an unclear mathematical relationship, and they appear to not be compatible with scalar multiplication. Because of this, they require extra bits to resolve ambiguity. There is only one known method for point compression in trace zero varieties over hyperelliptic curves from [Lan04b]. This method can be made practical for the parameters $g = 2, n = 3$.

One advantage of our representation with respect to the previous ones is that it is the only one that does not identify the positive and negative of a point, thus allowing a recovery of the v-polynomial of a compressed point that does not require computing square roots. For small values of n, this gives a noticeable advantage in efficiency. In addition, our method works for all affine points on the trace zero variety, without having to disregard a closed subset as it is done in [Sil05, Lan04b]. Furthermore, our compression and decompression algorithms do not require a costly precomputation, such as that of the Semaev polynomial in Chapter 4 or the elimination of variables from a polynomial system in [Lan04b].

In terms of efficiency, our compression algorithm is faster than all the other ones for elliptic curves except for our algorithm from Chapter 4, and our decompression algorithm is faster in all cases. For $g = 1$ and $n = 3, 5$, the time for compression and decompression together is comparable for $n = 3$, and smaller for $n = 5$, than that of Chapter 4. That is to say, the faster decompression makes up for the slower compression. Although here we concentrate on the case of odd characteristic, our method can be adapted to fields of even characteristic, just like all other methods from [Sil05, Lan04b, Nau99] and Chapter 4.

A detailed comparison of efficiency for elliptic curves is given in Chapter 6.4, so for the rest of this section we consider only hyperelliptic curves of genus $g \geq 2$. For $g = 2$ and $n = 3$, we compare the efficiency of our method with that of [Lan04b].

We have implemented all our algorithms in Magma [BCP97], and we give some timings to show how our implementation performs in practice. Notice however that our programs are straightforward implementations of the methods described here, and they are only meant as a proof of concept. Absolute timings with nothing to compare them to are not very significant, and we include them simply to be consistent with Chapter 6, where we compare our method for elliptic curves with the method from Chapter 4 on the basis of our own Magma implementations. In such a relative context, the timings are more meaningful.

All computations were done with Magma version 2.19.3 [BCP97], running on one core of an Intel Xeon X7550 Processor (2.00 GHz) on a Fujitsu Primergy RX900S1. Our timings are average values for one execution of the algorithm, where averages are computed over 10000 executions with random inputs.

Comparison and Timings for $g = 2, n = 3$. We present timings for trace zero subgroups of 20, 30, 40, 50, and 60 bits in Table 5.1. The reason for testing only such small groups is that it is difficult to produce larger ones in Magma without writing dedicated code. Since our implementation serves mostly as a proof of concept, and since this is not the focus of our work, we did not put much effort into producing suitable curves for larger trace zero subgroups. Avanzi and Cesena report in [AC07] that they were able to produce trace zero subgroups of 160 and 190 bits for curves of genus 2 over fields of even characteristic by modifying a software package written by Frederik Vercauteren.

The representation of [Lan04b] consists of 4 (out of 6) Weil restriction coordinates of the coefficients of the u-polynomial of a point plus two small numbers to resolve ambiguity. Following the notation of the original paper, we call the transmitted coordinates $u_{12}, u_{11}, u_{10}, u_{02}$, the two small numbers a, b, and the dropped coordinates u_{01}, u_{00}. This approach requires as a precomputation the elimination of 4 variables from a system of 6 equations of degree 3 in 10 variables. The result is a triangular system of 2 equations in 6 indeterminates. The compression algorithm plugs the values of $u_{12}, u_{11}, u_{10}, u_{02}$ into the system and solves for the two missing values in order to determine a, b, which in turn determine the roots coinciding with u_{01}, u_{00}. The decompression algorithm uses a, b to decide which among the solutions of the system are the coordinates it recovers. The advantage of this algorithm is that it works entirely over $\mathbb{F}_q$. Nevertheless, compression is much less efficient than our compression algorithm, since we only need to evaluate a number of expressions (and we do not need to compute roots or factor polynomials), while Lange has to solve a triangular system, which involves computing roots. While our decompression algorithm requires the factorization of one or two polynomials, which has complexity $O(\log q)$, Lange's decompression algorithm solves again the same triangular system. Since

Table 5.1 Average time in milliseconds for compression/decompression of one point when $g = 2, n = 3$

q	$2^5 - 1$	$2^8 - 75$	$2^{10} - 3$	$2^{13} - 2401$	$2^{15} - 19$
Compression	0.10	0.11	0.19	0.19	0.17
Full decompression	0.28	4.78	19.87	3.07	3.82

Table 5.2 Average time in milliseconds for compression/decompression of one point when $g \geq 7, n = 3$, $\log_2 |T_n| \approx 100$

g	7	8	9	10	11
q	$2^8 - 117$	$2^7 - 55$	$2^6 - 21$	$2^5 - 1$	$2^5 - 13$
Compression	3.81	0.92	0.93	1.22	2.07
Full decompression	5.46	27.39	5.03	6.81	5.93
g	12	13	14	15	16
q	$2^4 - 3$	$2^4 - 3$	$2^3 - 1$	$2^3 - 1$	$2^3 - 1$
Compression	2.08	2.04	6.46	8.60	3.67
Full decompression	18.32	19.73	13.33	12.79	13.45

Table 5.3 Average time in milliseconds for compression/decompression of one point when $g \geq 5, n = 5$, $\log_2 |T_n| \approx 160$

g	5	6	7	8	9	10	11
q	$2^8 - 5$	$2^7 - 27$	$2^6 - 23$	$2^5 - 1$	$2^4 - 5$	$2^4 - 5$	$2^4 - 5$
Compression	6.53	7.48	9.89	11.83	1.90	2.93	3.24
Full decompression	4.35	13.91	12.61	10.27	29.30	33.83	42.97

this involves computing roots in $\mathbb{F}_q$, which has complexity $O(\log^4 q)$ using standard methods (and can be as low as $O(\log^2 q)$ for special choices of parameters, see [BV06]), it is less efficient than the decompression algorithm proposed in this chapter. Notice also that Lange's approach does not give the v-polynomial, which needs to be computed separately, adding to the complexity of decompression.

Timings for $g > 2, n > 3$. As a proof of concept, we provide timings in Table 5.2 for trace zero subgroups of approximately 100 bits when $n = 3$ and $g = 7, 8, \dots, 16$ and in Table 5.3 for trace zero subgroups of approximately 160 bits when $n = 5$ and $g = 5, 6, \dots, 11$. The reason for this choice is simply that we are able to find suitable curves for these parameters. We stress again that the limitation here is not our compression method but finding trace zero subgroups of known group order, so we expect that our method will work for much larger values of n and g (e.g. we are able to compute an example for $g = 2, n = 23$, where the group has 173 bits).

5.6 Conclusions

In this chapter, we propose a representation of elements of the trace zero subgroup via rational functions. This representation is the only one (to the extent of our knowledge) that works for elliptic and hyperelliptic curves of any genus and field extensions of any prime degree. Our representation has convenient mathematical properties: It identifies well-defined equivalence classes of points, it is compatible with scalar multiplication, and it does not discard the v-polynomial of the Mumford representation (or the y-coordinate of an elliptic curve point), thus avoiding expensive square root computations in the decompression process.

Our compression and decompression algorithms are efficient, even for medium to large values of n and g. For those parameters where other compression methods are available (namely, for very small n and g), our algorithms are comparable with or more efficient than the previously known ones. No costly precomputation is required during the setup of the system.

Our optimal-sized and efficiently-computable representation, together with previous results on the security and on efficient arithmetic, make trace zero subgroups a very interesting class of groups in the context of public key cryptography.

Chapter 6

An optimal representation via rational functions – elliptic curves

We now specialize the results of Chapter 5 to elliptic curves, since elliptic curves are simpler and better studied than hyperelliptic curves. In particular, the Picard group of an elliptic curve is isomorphic to the curve itself. Therefore one can work with the group of points of the curve, and point addition is given by simple, explicit formulas. As we will see, it is also much easier to find a rational function with a given principal divisor. For all these reasons, the results and methods from Chapter 5 can be simplified and made explicit for elliptic curves.

Assumptions and Notation. *Throughout this chapter, let*

$$E : y^2 = f(x)$$

denote an elliptic curve defined over $\mathbb{F}_q$. The trace zero subgroup T_n of $E(\mathbb{F}_{q^n})$ is the group of all points P with trace equal to zero. In this chapter we consider only $n \geq 3$, since the case $n = 2$ is rather trivial, as explained at the beginning of Section 5.1. As in Chapter 5, we assume that $\mathbb{F}_q$ does not have characteristic 2, see Remark 5.4.

Roadmap. The organization of this chapter is analogous to that of Chapter 5: We introduce the representation in Section 6.1 and give the corresponding compression and decompression algorithms in Section 6.2. We present explicit equations in Section 6.3 and experimental results as well as a comparison with other representations in Section 6.4.

6.1 An optimal representation via rational functions

We have an analogous but more explicit version of Theorem 5.3. Although almost all statements follow from Theorem 5.3, we give direct and more elementary proofs below.

Notation 6.1. Write $P_i = \varphi^i(P)$ for $i = 0, \ldots, n-1$. Let

$$\ell_j(x, y) = 0, \quad j = 1, \ldots, n-2,$$

be the equation of the line passing through the points $P_0 \oplus \ldots \oplus P_{j-1}$ and P_j. We follow the usual convention that the line passing through P with multiplicity two is the tangent line to the curve at P. Let

$$v_j(x, y) = 0, \quad j = 1, \ldots, n-3,$$

be the equation of the vertical line passing through the point $P_0 \oplus \ldots \oplus P_j$.

Theorem 6.2. *Let $n \geq 3$ prime. For any $P \in T_n \setminus \{\mathcal{O}\}$, let*

$$h_P = \frac{\ell_1 \cdot \ldots \cdot \ell_{n-2}}{v_1 \cdot \ldots \cdot v_{n-3}} \in \mathbb{F}_q(E),$$

where ℓ_j and v_j are the lines defined in Notation 6.1. Then:

(*i*) *None of the lines $\ell_1, \ldots, \ell_{n-2}$ are vertical.*

(*ii*) $\operatorname{div}(h_P) = P_0 + \ldots + P_{n-1} - n\mathcal{O}$.

(*iii*) *$h_P(x,y) = h_{P,1}(x) + y h_{P,2}(x)$ for some $h_{P,1}, h_{P,2} \in \mathbb{F}_q[x]$.*

(*iv*) *$\deg h_{P,1} \leq \frac{n-1}{2}$ and $\deg h_{P,2} = \frac{n-3}{2}$.*

(*v*) *Let $h_{P,2}$ be monic, then h_P is uniquely determined by $n-1$ coefficients in $\mathbb{F}_q$.*

(*vi*) *$H_P = h_{P,1}^2 - f h_{P,2}^2$ has degree n, and its zeros are exactly the x-coordinates of $P_0, \ldots, P_{n-1}$.*

(*vii*) *If Q is such that $h_P = h_Q$, then $Q = \varphi^j(P)$ for some $j \in \{0, \ldots, n-1\}$.*

(*viii*) *$h_{P,2}(X) \neq 0$ for all x-coordinates X of $P_0, \ldots, P_{n-1}$.*

Proof. (*i*) Assume on the contrary that at least one of the lines is vertical. Let $j \in \{2, \ldots, n-1\}$ be minimal such that ℓ_{j-1} is vertical. Since ℓ_{j-1} passes through $P_0 \oplus \ldots \oplus P_{j-2}$ and P_{j-1}, this implies $P_0 \oplus \ldots \oplus P_{j-2} = w(P_{j-1})$, or $P_0 \oplus \ldots \oplus P_{j-1} = \mathcal{O}$. By applying φ^j, we may "shift" this equation and obtain $P_{mj} \oplus \ldots \oplus P_{(m+1)j-1} = \mathcal{O}$ for all $m \geq 0$ such that $(m+1)j \leq n$. We plug this into the trace zero equation $P_0 \oplus \ldots \oplus P_{n-1} = \mathcal{O}$ and apply an appropriate power of φ to get $P_0 \oplus \ldots \oplus P_{(n \bmod j)-1} = \mathcal{O}$. Now we start from the two smaller equations $P_0 \oplus \ldots \oplus P_{j-1} = \mathcal{O}$ and $P_0 \oplus \ldots \oplus P_{(n \bmod j)-1} = \mathcal{O}$ and apply an analogous procedure to further reduce the number of points in the sum. Similarly to the Euclidean Algorithm, we can keep reducing the larger index modulo the smaller one. Finally, we obtain an equation of the form $P_0 \oplus \ldots \oplus P_{\gcd(n,j)-1} = \mathcal{O}$. Since n is prime, we have $P_0 = \mathcal{O}$, a contradiction.

(*ii*) We have $\operatorname{div}(\ell_j) = (P_0 \oplus \ldots \oplus P_{j-1}) + P_j + w(P_0 \oplus \ldots \oplus P_j) - 3\mathcal{O}$ for $j \in \{1, \ldots, n-2\}$ and $\operatorname{div}(v_j) = (P_0 \oplus \ldots \oplus P_j) + w(P_0 \oplus \ldots \oplus P_j) - 2\mathcal{O}$ for $j \in \{1, \ldots, n-3\}$. Now we compute

$$\begin{aligned}
\operatorname{div}\left(\frac{\ell_1 \cdot \ldots \cdot \ell_{n-2}}{v_1 \cdot \ldots \cdot v_{n-3}}\right) &= \sum_{j=1}^{n-2} \operatorname{div}(\ell_j) - \sum_{j=1}^{n-3} \operatorname{div}(v_j) \\
&= P_0 + P_1 + \ldots + P_{n-2} + w(P_0 \oplus \ldots \oplus P_{n-2}) - n\mathcal{O} \\
&= P_0 + P_1 + \ldots + P_{n-1} - n\mathcal{O},
\end{aligned}$$

where $w(P_0 \oplus \ldots \oplus P_{n-2}) = P_{n-1}$ since P is a trace zero point.

(*iii*) For $i \in \{2, \ldots, n-1\}$, let

$$h_P^{(i)}(x,y) = \frac{\ell_1 \cdot \ldots \cdot \ell_{i-1}}{v_1 \cdot \ldots \cdot v_{i-2}}(x,y) \in \mathbb{F}_{q^n}(E).$$

We first show that $h_P^{(i)}(x,y) = h_{P,1}^{(i)}(x) + y h_{P,2}^{(i)}(x)$ for some polynomials $h_{P,1}^{(i)}$ and $h_{P,2}^{(i)}$ in $\mathbb{F}_{q^n}[x]$ by induction on i. The base case $i = 2$ is easy: $h_P^{(2)} = \ell_1$ is a line. For the induction step, we have $h_P^{(i)} = h_P^{(i-1)} \frac{\ell_{i-1}}{v_{i-2}}$, hence we need to show that v_{i-2} divides $h_P^{(i-1)} \ell_{i-1}$ in

$\overline{\mathbb{F}}_q[x,y]/(y^2-f(x))$. We write $P_0 \oplus \ldots \oplus P_{i-2} = (X,Y)$. Since v_{i-2} is the vertical line passing through (X,Y), we have $v_{i-2}(x,y) = x - X$. Hence to prove our claim, we check that $h_P^{(i-1)}\ell_{i-1} = 0$ in $\overline{\mathbb{F}}_q[x,y]/(y^2-f(x), x-X)$. In this ring, we have

$$\begin{aligned} h_P^{(i-1)}(x,y)\ell_{i-1}(x,y) &= (h_{P,1}^{(i-1)}(x) + yh_{P,2}^{(i-1)}(x))\ell_{i-1}(x,y) \\ &= (h_{P,1}^{(i-1)}(X) + yh_{P,2}^{(i-1)}(X))\ell_{i-1}(X,y) \\ &= c(y+Y)(y-Y) = c(y^2-Y^2) \\ &= c(f(x)-Y^2) = c(f(X)-Y^2) = 0 \end{aligned}$$

for some constant $c \neq 0$. The first equality is the induction hypothesis. For the third equality, we use that $h_P^{(i-1)}(X,-Y) = 0$ and $h_P^{(i-1)}(x,y)$ is linear in y, and that $\ell_{i-1}(X,Y) = 0$ and $\ell_{i-1}(x,y)$ is linear in y. In other words, we have a product of two linear polynomials in y, i.e. a quadratic polynomial in y, with zeros $\pm Y$. Therefore, the polynomial must be equal to $(y+Y)(y-Y)$ up to multiplication by a constant c.

The claim now follows for $i = n-1$, where $h_{P,1}, h_{P,2} \in \mathbb{F}_q[x]$ since $h_P \in \mathbb{F}_q(E)$ (recall that $\mathrm{Tr}(P)$ is $\mathbb{F}_q$-rational).

(iv) An easy induction on i, starting with $h_P^{(2)} = \ell_1$ a line and $h_P^{(3)} = \ell_1\ell_2/v_1$, shows that

$$\deg h_{P,1}^{(i)} \leq \begin{cases} \frac{i+1}{2} \text{ if } i \text{ is odd} \\ \frac{i}{2} \text{ if } i \text{ is even} \end{cases} \quad \text{and} \quad \deg h_{P,2}^{(i)} \leq \begin{cases} \frac{i-3}{2} \text{ if } i \text{ is odd} \\ \frac{i-2}{2} \text{ if } i \text{ is even.} \end{cases}$$

Then the inequalities $\deg h_{P,1} \leq \frac{n-1}{2}$ and $\deg h_{P,2} \leq \frac{n-3}{2}$ follow for $i = n-1$, since $n-1$ is even. Equality for the degree of $h_{P,2}$ follows from (i): Since the ℓ_j are never vertical, they have equations of the form $y + \mu_1 x + \mu_0$. Hence the numerator of $\frac{\ell_1\cdots\ell_{n-2}}{v_1\cdots v_{n-3}}$ has a term $y^{n-2} = yy^{n-3} = yx^{3(n-3)/2}$. The denominator is a polynomial in x of degree $n-3$. Therefore, the leading term of $h_{P,2}$ is $x^{3(n-3)/2-(n-3)} = x^{(n-3)/2}$.

(v) This follows directly from (iv).

(vi) The degree of $h_{P,1}^2$ is at most $n-1$, while the degree of $h_{P,2}^2$ is exactly $n-3$. This implies that H_P has degree n. We have $H_P(x) = (h_{P,1}(x) + yh_{P,2}(x))(h_{P,1}(x) - yh_{P,2}(x))$ as functions on E. Now from looking at the divisor (see (ii)), we see that the affine zeros of $h_{P,1}(x)+yh_{P,2}(x)$ on E are $P_0, \ldots, P_{n-1}$, and hence the affine zeros of $h_{P,1}(x)-yh_{P,2}(x)$ on E are $w(P_0), \ldots, w(P_{n-1})$. Therefore, the affine zeros of $H_P(x)$ on E are $\pm P_0, \ldots, \pm P_{n-1}$. Now since $H_P(x)$ has degree n, its zeros (as a polynomial) are exactly the x-coordinates of those points.

(vii) If $h_P = h_Q$, then $\mathrm{div}(h_P) = \mathrm{div}(h_Q)$, i.e. $\mathrm{Tr}(P) = \mathrm{Tr}(Q)$ and therefore P and Q must be Frobenius conjugates.

$(viii)$ Since $h_{P,2}$ is defined over $\mathbb{F}_q$, it suffices to show $h_{P,2}(X) \neq 0$ for the x-coordinate of P. By (iv) and (vi), $h_{P,2}$ is not the zero polynomial, since otherwise $\deg H_P = 2\deg h_{P,1} \leq n-1$. If $n = 3$ then $\deg h_{P,2} = 0$ by (iv), hence $h_{P,2}(X)$ is a non-zero constant for every X. Now suppose $n > 3$ and $h_{P,2}(X) = 0$, then $h_P(X,Y) = h_P(X,-Y) = 0$ and it follows that $w(P) = \varphi^j(P)$ for some j (this is analogous to Theorem 5.3 (vi)). If $j \neq 0$, then $X \in \mathbb{F}_{q^n} \cap \mathbb{F}_{q^j} = \mathbb{F}_q$ and $Y \in \mathbb{F}_{q^n} \cap \mathbb{F}_{q^{2j}} = \mathbb{F}_q$. Hence $P = w(P)$, i.e. P and all its Frobenius conjugates are 2-torsion points. On the other hand, $P \in T_n$ implies

$$P \oplus \varphi(P) \oplus \ldots \oplus \varphi^{n-1}(P) = \mathcal{O}.$$

Now if $P \in E(\mathbb{F}_q)$ then this implies $P \in E[n] \cap E[2] = \{\mathcal{O}\}$. If $P \notin E(\mathbb{F}_q)$ then the Frobenius conjugates of P are all distinct, a contradiction for $n > 3$ since E has only three 2-torsion points. $\square$

Since the exact degree of $h_{P,2}$ is known, h_P can be normalized by making $h_{P,2}$ monic. Hence one obtains an optimal representation for trace zero points on an elliptic curve.

Corollary 6.3. *Let $n \geq 3$ prime, and let $d_1 = (n-1)/2, d_2 = (n-3)/2$. Write $h_{P,1} = \gamma_{d_1} x^{d_1} + \ldots + \gamma_0$ and $h_{P,2} = x^{d_2} + \beta_{d_2-1} x^{d_2-1} + \ldots + \beta_0$. Define*

$$\begin{array}{rcl} \mathcal{R} : T_n \setminus \{\mathcal{O}\} & \longrightarrow & \mathbb{F}_q{}^{n-1} \\ P & \longmapsto & (\gamma_0, \ldots, \gamma_{d_1}, \beta_0, \ldots, \beta_{d_2-1}). \end{array}$$

Then $\mathcal{R}$ is an optimal representation for the elements of $T_n \setminus \{\mathcal{O}\}$ and

$$\mathcal{R}^{-1}(\mathcal{R}(P)) = \{P, \varphi(P), \ldots, \varphi^{n-1}(P)\} \textit{ for all } P \in T_n \setminus \{\mathcal{O}\}.$$

6.2 Compression and decompression algorithms

We detail in Algorithms 6.1 and 6.2 the simplified compression and decompression algorithms for elliptic curves. It is obvious that they are much shorter and simpler than the algorithms for hyperelliptic curves (see Algorithms 5.2 and 5.3).

Algorithm 6.1 Compression for elliptic curves, $n \geq 3$

```
Input: $P \in T_n$
Output: representation $(\alpha_0, \ldots, \alpha_{n-2}) \in \mathbb{F}_q^{n-1}$ of $P$
1: compute $h_P(x,y) = h_{P,1}(x) + y h_{P,2}(x) \leftarrow \frac{\ell_1 \cdot \ldots \cdot \ell_{n-2}}{v_1 \cdot \ldots \cdot v_{n-3}}(x,y)$ (see Algorithm 6.3) where
2: $h_{P,1}(x) = \gamma_{d_1} x^{d_1} + \ldots + \gamma_0$ and
3: $h_{P,2}(x) = x^{d_2} + \beta_{d_2-1} x^{d_2-1} + \ldots + \beta_0$
4: return $(\gamma_0, \ldots, \gamma_{d_1}, \beta_0, \ldots, \beta_{d_2-1})$
```

Algorithm 6.2 Decompression for elliptic curves, $n \geq 3$

```
Input: $(\alpha_0, \ldots, \alpha_{n-2}) \in \mathbb{F}_q^{n-1}$
Output: one point $P \in T_n \setminus \{\mathcal{O}\}$ with representation $(\alpha_0, \ldots, \alpha_{n-2})$
1: $h_{P,1}(x) \leftarrow \alpha_{(n-1)/2} x^{(n-1)/2} + \alpha_{(n-3)/2} x^{(n-3)/2} + \ldots + \alpha_1 x + \alpha_0$
2: $h_{P,2}(x) \leftarrow x^{(n-3)/2} + \alpha_{n-2} x^{(n-5)/2} + \ldots + \alpha_{(n+3)/2} x + \alpha_{(n+1)/2}$
3: $H_P(x) \leftarrow h_{P,1}(x)^2 - f(x) h_{P,2}(x)^2$
4: $X \leftarrow$ one root of $H_P(x)$
5: $Y \leftarrow -h_{P,1}(X)/h_{P,2}(X)$
6: return $P = (X, Y)$
```

Finally, we discuss how to compute h_P for a given $P \in T_n$. Explicit formulas can be computed in the special cases $n = 3, 5$. We do this in the next section. For general n, a straightforward computation of h_P is possible, since we have an explicit formula given in terms of lines whose equations we know. Such a computation can be made more efficient by employing the usual divide and conquer strategy. Computing h_P via a Miller-style algorithm analogous to Algorithm 5.1 is also possible. The latter is advantageous for medium and large values of n, while for small values of n (for which we do not have explicit formulas) a straightforward computation using a divide and conquer approach seems preferable.

In Algorithm 6.3 we give a Miller-style algorithm to compute h_P. We denote by $\ell_{P,Q}$ the line through the points P and Q and by v_P the vertical line through P. All computations are done with functions on E, i.e. in $\mathbb{F}_{q^n}(E)$.

Algorithm 6.3 Miller-style double and add algorithm for computing $h_P, n \geq 3$

```
Input: $P \in T_n \setminus \{\mathcal{O}\}$ and $n - 1 = \sum_{j=0}^{s} n_j 2^j$
Output: $h_P$
 1: $Q \leftarrow \varphi(P)$
 2: $h \leftarrow \ell_{P,Q},\ R \leftarrow P \oplus Q,\ Q \leftarrow \varphi(Q),\ i \leftarrow 2$
 3: if $n_{s-1} = 1$ then
 4:     $h \leftarrow h \cdot \frac{\ell_{R,Q}}{v_R},\ R \leftarrow R \oplus Q,\ Q \leftarrow \varphi(Q),\ i \leftarrow 3$
 5: end if
 6: for $j = s-2, s-3, \ldots, 1, 0$ do
 7:     $h \leftarrow h \cdot h^{\varphi^i} \cdot \frac{v_{R+\varphi^i(R)}}{\ell_{w(R),w(\varphi^i(R))}},\ R \leftarrow R \oplus \varphi^i(R),\ Q \leftarrow \varphi^i(Q),\ i \leftarrow 2i$
 8:     if $n_j = 1$ then
 9:         $h \leftarrow h \cdot \frac{\ell_{R,Q}}{v_R},\ R \leftarrow R + Q,\ Q \leftarrow \varphi(Q),\ i \leftarrow i + 1$
10:     end if
11: end for
12: return $h$
```

Corollary 6.4. (*i*) *The execution of Algorithm 6.3, and therefore also of compression Algorithm 6.1, requires $O(3^{\log n})$ operations in $\mathbb{F}_{q^n}$.*

(*ii*) Decompress(Compress(P)) *is one of the Frobenius conjugates of P. The complexity of decompression Algorithm 6.2 is $O(n^{\log_2 3+1} \log n \log(nq))$ operations in $\mathbb{F}_{q^n}$.*

Proof. (*i*) See Theorem 5.9.

(*ii*) Theorem 5.11 (iii) would give a complexity of $O(n^{\log_2 3+2} \log(nq))$ operations in $\mathbb{F}_{q^n}$. However, the situation here is simpler, since we know that H_P must split into linear factors over $\mathbb{F}_{q^n}$. Therefore, we apply the root finding algorithm of [GvzG99, Algorithm 14.15], which has a better complexity of $O(n^{\log_2 3+1} \log n \log(nq))$ operations in $\mathbb{F}_{q^n}$. □

Remark 6.5. A more careful analysis of Algorithm 6.3 (using the bounds on the degrees of $h^{(i)}$ from the proof of Theorem 6.2 (iv)) shows that the compression complexity that we give in the previous corollary is not only an asymptotic one, but a rather precise operation count. Therefore, we can predict the behavior of the compression algorithm for relatively small values of n. In practice it behaves better than the obvious way of computing h_P (i.e. iteratively multiplying by $\frac{\ell_i}{v_{i-1}}$) for $n > 10$ and better than a divide and conquer approach for $n > 20$.

6.3 Explicit equations

For the simple cases, we give explicit equations for compression and decompression. In addition to making the computation more efficient, they allow us to perform precise operation counts and thus to compare our method to the other existing compression methods in Section 6.4.

6.3.1 Explicit equations for $g = 1, n = 3$

This case is particularly simple, since $h_P = \ell_1$ is just a line through the points $P, \varphi(P)$, and $\varphi^2(P)$. *For the sake of concreteness, we assume that $\mathbb{F}_q$ does not have characteristic 2 or 3 and that E is given by an equation in short Weierstraß form*

$$E : y^2 = x^3 + Ax + B.$$

Formulas and operation counts can easily be adjusted to the other cases. It will also be useful to choose a basis of the field extension $\mathbb{F}_{q^3}|\mathbb{F}_q$. *For simplicity, we also assume that* $3 \mid q-1$ *and write* $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - \mu)$ *as a Kummer extension*, where $\mu \in \mathbb{F}_q$ is not a third power. Then $1, \zeta, \zeta^2$ is a basis of $\mathbb{F}_{q^3}|\mathbb{F}_q$. It is highly likely that there exists a suitable μ of small size, see [Lan04b, Section 3.1].

Remark 6.6. When required to work with a field extension where $3 \nmid q-1$, one may choose a normal basis, which yields similar but dense equations, see Remark 3.1.

Compression. If $P = (X, Y) \notin E(\mathbb{F}_q)$, then the equation of $h_P = \ell_1$ is

$$h_P = y - \frac{Y^q - Y}{X^q - X}x + \frac{Y^q - Y}{X^q - X}X - Y = yh_{P,2}(x) + h_{P,1}(x).$$

Therefore, we have $h_{P,2}(x) = 1$ and $h_{P,1}(x) = \gamma_1 x + \gamma_0$, where

$$\begin{aligned}\gamma_1 &= -\frac{Y^q - Y}{X^q - X}\\ \gamma_0 &= -\gamma_1 X - Y,\end{aligned}$$

and Compress$(P) = (\gamma_0, \gamma_1)$.

Since γ_0, γ_1 are in $\mathbb{F}_q$, it is more efficient to carry out the entire computation in $\mathbb{F}_q$. Using the basis $1, \zeta, \zeta^2$ of $\mathbb{F}_{q^3}|\mathbb{F}_q$, we write

$$\begin{aligned}X &= X_0 + X_1\zeta + X_2\zeta^2\\ Y &= Y_0 + Y_1\zeta + Y_2\zeta^2.\end{aligned} \tag{6.1}$$

Then γ_0 and γ_1 can be computed directly from $X_0, X_1, X_2, Y_0, Y_1, Y_2$ over $\mathbb{F}_q$ as

$$\begin{aligned}\gamma_1 &= \frac{c_1 X_1^2 Y_1 + c_2 X_2^2 Y_2}{c_1 X_1^3 + c_2 X_2^3}\\ \gamma_0 &= -\gamma_1 X_0 - Y_0,\end{aligned}$$

where

$$\begin{aligned}c_1 &= 1 - \mu^{(q-1)/3}\\ c_2 &= \mu^{1+(q-1)/3} - \mu = -\mu c_1\end{aligned}$$

are constants and can be precomputed during the setup phase of the algorithm.

When $P \in E(\mathbb{F}_q)$, the line ℓ_1 is a tangent and we have

$$\begin{aligned}\gamma_1 &= \frac{3X^2 + A}{2Y}\\ \gamma_0 &= -\gamma_1 X - Y.\end{aligned}$$

Notice that such points are in $E[3](\mathbb{F}_q)$ and therefore very few. In the general case $P \notin E(\mathbb{F}_q)$, compression takes 2S+6M+1I in $\mathbb{F}_q$.

Decompression. This algorithm computes the polynomial H_P and its roots over $\mathbb{F}_{q^3}$. We have

$$H_P(x) = x^3 - S_1 x^2 + S_2 x - S_3$$

where

$$\begin{aligned} S_1 &= \gamma_1^2 \\ S_2 &= A - 2\gamma_0\gamma_1 \\ S_3 &= \gamma_0^2 - B. \end{aligned}$$

Computing the coefficients of H_P therefore takes 2S+1M in $\mathbb{F}_q$. Since the roots of this polynomial are X, X^q, X^{q^2}, and using (6.1), we get

$$\begin{aligned} S_1 &= X + X^q + X^{q^2} &&= 3X_0 \\ S_2 &= X^{1+q} + X^{1+q^2} + X^{q+q^2} &&= 3X_0^2 - 3\mu X_1X_2 \\ S_3 &= X^{1+q+q^2} &&= X_0^3 - 3\mu X_0X_1X_2 + \mu X_1^3 + \mu^2 X_2^3. \end{aligned}$$

Hence one can solve the system

$$\begin{aligned} S_1 &= 3x_0 \\ S_2 &= 3x_0^2 - 3\mu x_1x_2 \\ S_3 &= x_0^3 - 3\mu x_0x_1x_2 + \mu x_1^3 + \mu^2 x_2^3 \end{aligned}$$

over $\mathbb{F}_q$ to recover (X_0, X_1, X_2). Since the solutions of the system are exactly the Frobenius conjugates of X via (6.1), it suffices to find a single solution. This takes at most 3S+3M+1I, one square root, and two cube roots in $\mathbb{F}_q$ (see Chapter 4.5.1). Notice that, since this system is so simple, this is more efficient than factoring the polynomial H_P over $\mathbb{F}_{q^3}$.

Finally, $Y = -\gamma_1 X - \gamma_0$, so recomputing one y-coordinate takes 1M in $\mathbb{F}_q$, and the other ones can be recovered via the Frobenius map.

In total, decompression takes at most 5S+5M+1I, one square root, and two cube roots in $\mathbb{F}_q$.

6.3.2 Explicit equations for $g = 1, n = 5$

We also give explicit formulas for elliptic curves and field extensions of degree 5. Our experimental results show that these formulas are the fastest way to compress and decompress points, and we use them in our implementation, as discussed in Section 6.4. *Again, we assume that E is given in short Weierstraß form $E : y^2 = x^3 + Ax + B$ over a field of characteristic not equal to 2 or 3.*

Compression. Let $P = (X, Y) \in T_5$, and denote by $\lambda_1, \lambda_2, \lambda_3$ the slopes of the lines ℓ_1, ℓ_2, ℓ_3, respectively. We have

$$h_P = \frac{\ell_1\ell_2\ell_3}{v_1v_2} = (\gamma_2 x^2 + \gamma_1 x + \gamma_0) + y(x + \beta_0),$$

where

$$\begin{aligned} \gamma_2 &= -\lambda_1 - \lambda_2 - \lambda_3 \\ \beta_0 &= -\lambda_2\gamma_2 + \lambda_1\lambda_3 - X^{q^2} \\ \gamma_1 &= -\lambda_2\beta_0 - \gamma_2 X^{q^2} + \lambda_1 X + \lambda_3 X^{q^3} - Y - Y^{q^2} - Y^{q^3} \\ \gamma_0 &= \gamma_1(\lambda_2^2 - X^{q^2}) + \gamma_2((X + X^q)(X + X^q - X^{q^2} - 2\lambda_1^2 + \lambda_2^2) + \lambda_1^4 + A + \lambda_1^2 X^{q^2}) \\ &\quad + \lambda_1\lambda_2\lambda_3(X + X^{q^2} + X^{q^3}) - \lambda_1\lambda_2 Y^{q^3} - \lambda_1\lambda_3 Y^{q^2} - \lambda_2\lambda_3 Y \\ &\quad + \lambda_3\lambda_1^2\lambda_2^2 + \lambda_1^3\lambda_2^2 + \lambda_1^2\lambda_2^3. \end{aligned}$$

Here, we do arithmetic in $\mathbb{F}_{q^5}$ and therefore count operations in $\mathbb{F}_{q^5}$. Computing $\lambda_1, \lambda_2, \lambda_3$ takes a total of 3M+3I. Then, $\beta_0, \gamma_0, \gamma_1, \gamma_2$ can be computed with a total of 3S+15M. Thus, compression takes a total of 3S+18M+3I in $\mathbb{F}_{q^5}$.

Decompression. We compute

$$\begin{aligned} S_1 &= \gamma_2^2 - 2\beta_0 \\ S_2 &= \beta_0^2 + A - 2\gamma_1\gamma_2 \\ S_3 &= \gamma_1^2 + 2\gamma_0\gamma_2 - 2A\beta_0 - B \\ S_4 &= A\beta_0^2 + 2B\beta_0 - 2\gamma_0\gamma_1 \\ S_5 &= \gamma_0^2 - B\beta_0^2 \end{aligned}$$

using 4S+3M in $\mathbb{F}_q$. Then we factor the polynomial $H_P(x) = x^5 - S_1x^4 + S_2x^3 - S_3x^2 + S_4x - S_5$, which takes $O(\log q)$ operations in $\mathbb{F}_q$. Finally, recovering Y costs 1S+3M+1I in $\mathbb{F}_{q^5}$.

6.4 Timings and comparison with other representations

A general discussion about how this compression method compares to the other known approaches is given in Chapter 5.5, including the case of elliptic curves. Here, we compare the efficiency of our algorithms with those of Chapter 4 and [Sil05, Nau99] in more detail. The comparison of our method with that of Chapter 4 is on the basis of a precise operation count, complexity analysis, and our own Magma implementations. Our timings are average values for one execution of the algorithm, where averages are computed over 10000 executions with random inputs. Our comparison with [Nau99, Sil05] is rougher, since no precise operation counts, complexity analyses or implementations of those methods are available. Nevertheless, our analysis leads to a meaningful comparison of efficiency in all cases.

Comparison and Timings for $g = 1, n = 3$. We compare our method with the better method of Chapter 4 (there called "compression in t_i") in terms of operations in Table 6.1 and timings in Table 6.2, where we highlight the best (i.e. smallest) times in italics. We choose arbitrary elliptic curves such that the associated trace zero subgroups have prime order for fields of 20, 40, 60, and 79 bits. We see that the compression algorithm from Chapter 4 requires fewer operations, but not in a significant way. These small differences are obviously not measured accurately in our tests. Our measurements for decompression are more meaningful, however. We compare "full decompression", where one entire point (including the y-coordinate) is recomputed. Here, the method of Chapter 4 is much slower (roughly a factor 10), due to the necessary square root extraction. This shows one major efficiency advantage of the approach that we follow in this chapter: Recovering the y-coordinate is much faster, since no square root computation is necessary. For a different point of view, we also compare "decompression in x only", where no y-coordinate is computed. In this case, the algorithm proposed in this chapter and the one from Chapter 4 behave similarly.

Naumann's method [Nau99], as explained in Chapter 4.1, requires the factorization of a degree 4 polynomial for both compression and decompression. For decompression, the y-coordinate must then be recomputed as a square root. This is clearly more expensive than the decompression algorithm presented in this chapter, which does not require polynomial factorization or square root extraction. Our compression algorithm is also more efficient, since it computes only some very simple expressions over $\mathbb{F}_q$.

Table 6.1 Number of operations in $\mathbb{F}_q$ for compression/decompression of one point when $n = 3$, smallest values in italics

Compression	2S+6M+1I
Compression Chapter 4	*1M*
Full decompression	5S+5M+1I, 1 square root, 2 cube roots
Full decompression Chapter 4	4S+3M+2I, 1 sqrt, 2 cube roots, and 1 sqrt in $\mathbb{F}_{q^3}$
Decompression x only	*5S+4M+1I, 1 square root, 2 cube roots*
Decompression x only Chapter 4	4S+3M+2I, 1 square root, 2 cube roots

Table 6.2 Average time in milliseconds for compression/decompression of one point when $n = 3$, smallest values in italics

q	$2^{20} - 3$	$2^{40} - 87$	$2^{60} - 93$	$2^{79} - 67$
Compression	*0.01*	0.03	*0.03*	*0.04*
Compression Chapter 4	*0.01*	*0.02*	*0.03*	*0.04*
Full decompression	0.18	0.71	0.89	1.52
Full decompression Chapter 4	0.84	7.62	10.62	17.58
Decompression x only	*0.15*	*0.63*	*0.87*	*1.40*
Decompression x only Chapter 4	*0.15*	0.68	*0.87*	1.44

In [Sil05], compression requires computing the root of a degree 4 polynomial. As explained above, our compression algorithm is faster. The bulk of the work in the decompression algorithm is factoring a degree 4 polynomial and recomputing the y-coordinate from the curve equation. This is clearly more expensive than the decompression algorithm in this chapter. See Chapter 4.5.1 for a more detailed discussion of the decompression algorithm from [Sil05].

Comparison and Timings for $g = 1, n = 5$. A similar comparison for extension degree 5 (see Tables 6.3 and 6.4, best times in italics) shows that the compression algorithm proposed in this chapter is less efficient than that of Chapter 4, but the decompression algorithm is faster. Although the bulk of the work in both decompression algorithms is polynomial factorization, following the approach proposed in this chapter we have to factor one polynomial of degree 5 over $\mathbb{F}_{q^5}$, where the algorithm of Chapter 4 first factors a polynomial of degree 6 over $\mathbb{F}_q$ and then at least one polynomial of degree 5 over $\mathbb{F}_{q^5}$. For this reason, the decompression algorithm proposed in this chapter performs much better than that of Chapter 4, regardless of whether we include the recovery of the y-coordinate. Notice that we again compare with the best method from Chapter 4, there called "compression/decompression in the s_i with polynomial factorization".

In comparison to [Sil05], both our compression and our decompression methods are much more efficient. The algorithms of Silverberg involve resultant computations and the factorization of a degree 27 polynomial. For more detail, see Chapter 4.5.2.

Timings for $g = 1, n > 5$. We study the performance of our algorithms by means of experimental results for $n > 5$. First, for comparison with the last column of Tables 6.2 and 6.4, we give in Table 6.5 timings for $n = 7, 11, 13, 19, 23$ and corresponding randomly chosen values of q, A, and B that produce prime order trace zero subgroups of approximately 160 bits. From the different values for decompression times (due to the fact that the

Table 6.3 Operations/complexity for compression/decompression of one point when $n = 5$, smallest values in italics

Compression	3S+18M+3I in $\mathbb{F}_{q^5}$
Compression Chapter 4	*5S+13M in* $\mathbb{F}_q$
Full decompression	$O(\log q)$ *operations in* $\mathbb{F}_q$
Full decompression Chapter 4	$O(\log q)$ operations in $\mathbb{F}_q$, and 1 square root in $\mathbb{F}_{q^5}$
Decompression x only	$O(\log q)$ *operations in* $\mathbb{F}_q$
Decompression x only Chapter 4	$O(\log q)$ *operations in* $\mathbb{F}_q$

Table 6.4 Average time in milliseconds for compression/decompression of one point when $n = 5$, smallest values in italics

q	$2^{10} - 3$	$2^{20} - 5$	$2^{30} - 173$	$2^{40} - 195$
Compression	0.21	0.25	0.46	0.80
Compression Chapter 4	*0.04*	*0.04*	*0.05*	*0.10*
Full decompression	0.82	9.39	4.26	10.13
Full decompression Chapter 4	5.89	17.90	30.21	63.60
Decompression x only	*0.77*	*9.36*	*4.01*	*9.82*
Decompression x only Chapter 4	5.53	16.48	21.42	45.08

Table 6.5 Average time in milliseconds for compression/decompression of one point when $n > 5$, $\log_2 |T_n| \approx 160$

n	7	11	13	19	23
q	$2^{27} - 27689095$	$2^{16} - 129$	$2^{14} - 6113$	$2^9 - 55$	$2^8 - 117$
Compression	1.80	2.84	3.89	8.82	12.90
Full decompression	20.90	10.16	4.03	119.75	58.15

performance of the polynomial factorization algorithm in Magma depends heavily on the specific choice of q and n), we see that there is much room for optimization in the choice of these parameters.

For a better overview on how our algorithms behave for larger n, we also give time measurements for trace zero subgroups of approximately 500 bits in Table 6.6. These groups are, of course, much larger than those used for cryptosystems in practice.

In each case, we choose the fastest method of computing h_P during compression. As discussed in Remark 6.5, this is an iterative approach for $n = 7$, a divide and conquer approach for $n = 11, 13, 19$, and Algorithm 6.3 for $n \geq 23$. During decompression we compute the y-coordinate of the point as well, since the difference with computing the x-coordinate only is negligible.

We also report that we are able to apply our method to much larger trace zero subgroups and much larger values of n. More specifically, our implementation works for trace zero subgroups of more than 3000 bits and for values of n larger than 300. For even larger values of n, the limitation is not our compression/decompression approach, but rather the fact that the trace zero subgroup becomes very large, even for small fields.

Table 6.6 Average time in milliseconds for compression/decompression of one point when $n > 5$, $\log_2 |T_n| \approx 500$

n	7	11	13
q	$2^{84} - 71576044106829976776257737$	$2^{50} - 431$	$2^{42} - 907314682077$
Compression	4.439	9.860	14.986
Full decompression	50.772	2539.269	959.298
n	19	23	29
q	$2^{28} - 38321553$	$2^{23} - 1445367$	$2^{18} - 24981$
Compression	16.123	13.884	22.250
Full decompression	264.201	18.471	103.363
n	31	37	41
q	$2^{17} - 27159$	$2^{14} - 1731$	$2^{13} - 2451$
Compression	25.938	36.818	45.191
Full decompression	768.097	526.804	116.639
n	42	47	53
q	$2^{12} - 483$	$2^{11} - 261$	$2^{10} - 281$
Compression	50.110	62.002	82.214
Full decompression	568.539	314.462	49.198

Chapter 7

An index calculus attack on the discrete logarithm problem

Let T_n be the trace zero variety of an elliptic curve E defined over $\mathbb{F}_q$. Then solving a DLP in $E(\mathbb{F}_{q^n})$ has the same complexity as solving a DLP in T_n (see Proposition 2.19). In other words, the DLP in $E(\mathbb{F}_{q^n})$ is as hard as the DLP in T_n, although T_n is a proper subgroup of $E(\mathbb{F}_{q^n})$. Given that the complexity of discrete logarithm algorithms often depends mainly on the group size, this means that it can be advantageous to attack the DLP in the smaller group T_n, rather than in $E(\mathbb{F}_{q^n})$. Moreover, T_n has been proposed for pairing-based cryptosystems, since it yields a particularly high security parameter, and for DLP-based cryptosystems, due to its efficient arithmetic. Therefore, we study the DLP in T_n in this chapter.

More specifically, we present an index calculus algorithm for $T_n = V_n(\mathbb{F}_q)$, following the approach of Gaudry [Gau09] for index calculus in abelian varieties (see Chapter 2.7) and making crucial use of the Semaev polynomials [Sem04] (see Chapter 3.3). The complexity of this attack depends on the size of $\mathbb{F}_q$ and the dimension of the variety: Asymptotically in q (and considering n to be a constant), it has complexity $\tilde{O}(q^{2-2/(n-1)})$, which is lower than that of generic attacks on T_n for $n > 3$ and lower than that of generic attacks on $E(\mathbb{F}_{q^n})$ for $n > 2$. This leads to the best known attack on the DLP in $E(\mathbb{F}_{q^n})$ for prime n. Here we demonstrate that the attack is feasible for $n = 3$ and q up to about 30 bits (then $E(\mathbb{F}_{q^3})$ is a 90-bit group). Our computations show that when $n = 3$, the index calculus attack is faster than a Pollard–Rho attack on $E(\mathbb{F}_{q^3})$ for $\log_2 q \geq 30$ approximately. The attack does not affect groups $E(\mathbb{F}_p)$ where p is a prime.

We also analyze the algorithm asymptotically in n and q, and we see that the complexity is exponential in n. This is mostly due to the fact that in order to produce relations, the algorithm has to solve polynomial systems whose size (number of equations, number of indeterminates, degrees of the equations) depends on n and that the Gröbner basis methods that we use to solve these systems have a large complexity in these parameters. We conclude that one can only hope to produce relations with this method for small values of n.

For the cases $n = 3, 5$ we write down explicitly the systems that one gets, and we experiment how to solve them with Magma [BCP97]. Based on this data, we discuss the practicality of the attack. For $n = 3$ we find that the systems can be solved easily and that the index calculus attack is hence feasible for q up to about 30 bits. For $n = 5$ we see that solving the polynomial system is extremely difficult already. Using some tricks (suggested in [BFP08, JV12]), we are able to produce relations and to solve a DLP for very small q, but the attack this yields is not better than generic attacks in overall complexity, therefore it is not a threat to the DLP in T_5 or $E(\mathbb{F}_{q^5})$.

Finally, we compare this algorithm to other existing attacks and draw some general conclusion on the DLP in T_n.

Assumptions and Notation. *In this chapter, let $T_n = V_n(\mathbb{F}_q)$ be the trace zero subgroup of an elliptic curve E over a finite field $\mathbb{F}_q$, where $n \geq 3$ is prime.*

Roadmap. We describe the application of Gaudry's algorithm to V_n in Section 7.1. Then we analyze its complexity in Section 7.2. In Section 7.3, we present explicit equations and Magma experiments for $n = 3, 5$. Finally, we compare the index calculus attack with other attacks on the DLP in T_n in Section 7.4 and discuss the implications of our results for trace zero elliptic curve cryptosystems (including pairing-based ones) in Section 7.5.

7.1 An index calculus algorithm for the trace zero variety

Background on index calculus in general and on Gaudry's algorithm [Gau09] in particular is given in Section 2.7. Following the ideas of Gaudry, we propose the following index calculus algorithm to compute discrete logarithms in T_n. When $n = 2$, then V_n is one-dimensional and the attack cannot be applied. Therefore, we only consider $n \geq 3$. Furthermore, we assume that T_n is cyclic, which is a common assumption in cryptography (it is true if $|T_n|$ is prime). For given q and n, curves E such that T_n is cyclic are easy to find, in fact in our experiments we did not encounter a single pair (q, n) where we could not find a suitable elliptic curve E.

Remark 7.1. When T_n is not cyclic, some of the probability estimates in Section 7.2 may be wrong and the algorithm may not function as expected. However, these problems can be overcome using classical randomization techniques (see [Gau09, Remark 2], [EG02]).

Remark 7.2. In the description and analysis of his algorithm, Gaudry often makes the assumption that the involved objects behave like "random" or generic ones, although in reality they are not random at all. Rather, they are specially constructed to have certain properties. For example, he defines the factor base as the $\mathbb{F}_q$-rational points of a curve, which he obtains by intersecting the d-dimensional variety with $d-1$ hyperplanes. Although this produces a one-dimensional subvariety generically, that may not be true in some cases. Furthermore, Gaudry assumes that the curve one obtains is absolutely irreducible and smooth, since then he may use the Theorem of Hasse–Weil to conclude that the number of $\mathbb{F}_q$-rational points on the curve is about q. As a justification for these assumptions, Gaudry suggests to "randomize coordinates", i.e. apply a random linear change of coordinates, as a first step of the algorithm. While this does not change the complexity of the algorithm, it implies that with high probability, the objects encountered during the algorithm behave like generic ones, or in other words, the probability that an undesired event happens is small. Whenever the algorithm runs into a case where the objects do not behave as desired, it starts over with a new change of coordinates. Therefore in our exposition of the algorithm for the trace zero variety, we also assume that the objects behave like generic objects, keeping in mind that otherwise, we may just apply a random change of coordinates and start over. However, our experiments suggest that this is not necessary and that the choices we describe work in many cases, in fact in all cases we have tried.

The algorithm takes as input two points $P, Q \in T_n$ such that $T_n = \langle P \rangle$, and it outputs the discrete logarithm $\log_P Q$, i.e. a number $\ell = \log_P Q \in \mathbb{Z}/\operatorname{ord}(P)\mathbb{Z}$ such that $[\ell]P = Q$ in T_n. Below, we describe the different steps of the algorithm in detail.

7.1.1 Setup

Following the suggestion of Semaev [Sem04], we carry out the index calculus algorithm working only with the x-coordinates of points in T_n. We choose a basis $\{1, \zeta, \ldots, \zeta^{n-1}\}$ of the extension $\mathbb{F}_{q^n}|\mathbb{F}_q$ and represent an affine point $P = (X, Y) \in T_n$ via the coordinates

$$P = (X_0, \ldots, X_{n-1}),$$

where $X = X_0 + X_1\zeta + \ldots + X_{n-1}\zeta^{n-1}$. So by writing $(X_0, \ldots, X_{n-1}) \in T_n$ we mean that there exists a Y such that $(X, Y) \in T_n$.

As in Chapter 3.3, denote by $f_n(z_1, \ldots, z_n)$ the n-th Semaev polynomial (see Definition 3.7). Then the Weil restriction of $f_n(x, x^q, \ldots, x^{q^{n-1}})$, using $x = x_0 + x_1\zeta + \ldots + x_{n-1}\zeta^{n-1}$ and reducing modulo $x_i^q - x_i$ for all i, yields one single equation, which we denote by $\tilde{f}_n(x_0, \ldots, x_{n-1})$. All points $P \in T_n$ satisfy $\tilde{f}_n(X_0, \ldots, X_{n-1}) = 0$, and conversely the $\mathbb{F}_q$-solutions $(X_0, \ldots, X_{n-1})$ of the equation

$$\tilde{f}_n(x_0, \ldots, x_{n-1}) = 0 \tag{7.1}$$

yield x-coordinates of points in T_n via $(X_0, \ldots, X_{n-1}) \mapsto X = X_0 + X_1\zeta + \ldots + X_{n-1}\zeta^{n-1}$, provided that the corresponding y-coordinates are in $\mathbb{F}_{q^n}$ and up to some exceptions (see Lemma 3.10). For $n = 3, 5$, these exceptions are well-understood and their number is very small (see Proposition 3.11 and Remark 3.12). Therefore, we henceforth use (7.1) as an equation for the trace zero subgroup. It has total degree $(n-1)2^{n-2}$.

7.1.2 Factor base

We define the factor base

$$\mathcal{F} = \{(0, \ldots, 0, X_{n-2}, X_{n-1}) \in T_n\}.$$

These are the $\mathbb{F}_q$-rational points of a curve in V_n obtained by intersecting V_n with the hyperplanes $\{x_0 = 0\}, \ldots, \{x_{n-3} = 0\}$. Since V_n has dimension $n-1$, intersecting with $n-2$ hyperplanes generically gives a subvariety of dimension 1. Thus $\mathcal{F}$ has about q elements by the Hasse–Weil Theorem, provided that this subvariety is absolutely irreducible.

Remark 7.3. Important properties of the factor base are that it has about q elements (this will be used in the complexity analysis, see Section 7.2) and that its elements can be described via algebraic equations (this will allow us to describe relations via a polynomial system, see Section 7.1.3). A further very important property is that the factor base must generate a large part of T_n, so that many elements of T_n decompose over the factor base. For this reason, the curve should not be contained in any proper abelian subvariety of V_n. Moreover, the fact that $|\mathcal{F}| \approx q$ can be proven (with the Theorem of Hasse–Weil) only if we assume that the curve is smooth and absolutely irreducible. Notice, however, that intersecting V_n with any choice of $n-2$ hyperplanes, or equivalently, setting any $n-2$ of the x_i equal to zero, may give a factor base with q elements. Therefore, if setting $x_0 = \ldots = x_{n-3} = 0$ happens to be a bad choice, we simply make a different one. In our current exposition we assume that the choice we have made is a good one. This is true in all our experiments.

Using equation (7.1), we see that any element $(X_0, \ldots, X_{n-1}) \in \mathcal{F}$ satisfies the equations

$$\begin{array}{rcl} \tilde{f}_n(X_0, \ldots, X_{n-1}) & = & 0 \\ X_0 & = & 0 \\ & \vdots & \\ X_{n-3} & = & 0, \end{array}$$

or, by plugging the last equations into the first one,

$$\tilde{f}_n(0, \ldots, 0, X_{n-2}, X_{n-1}) = 0.$$

Conversely, the $\mathbb{F}_q$-solutions (X_{n-2}, X_{n-1}) of

$$\tilde{f}_n(0, \ldots, 0, x_{n-2}, x_{n-1}) = 0 \tag{7.2}$$

yield x-coordinates of points in $\mathcal{F}$ via $(X_{n-2}, X_{n-1}) \mapsto X_{n-2}\zeta^{n-2} + X_{n-1}\zeta^{n-1}$, provided that the corresponding y-coordinates are in $\mathbb{F}_{q^n}$ and up to a few exceptions, as explained above (see also Lemma 3.10, Proposition 3.11, Remark 3.12).

7.1.3 Relation collection

Since V_n has dimension $n-1$, we search for relations of the form

$$R = P_0 \oplus \ldots \oplus P_{n-2}, \tag{7.3}$$

where $R = [\alpha]P \oplus [\beta]Q \in T_n$ is given and $P_0, \ldots, P_{n-2} \in \mathcal{F}$ are to be found. We write $U = U_0 + U_1\zeta + \ldots + U_{n-1}\zeta^{n-1}$ for the x-coordinate of R.

We use the Semaev polynomial to describe a relation. This is the purpose for which Semaev originally proposed his summation polynomials. If the points $P_0, \ldots, P_{n-2}$ with x-coordinates $X_{P_0}, \ldots, X_{P_{n-2}}$ are given, then according to Theorem 3.8 they satisfy (7.3) if and only if

$$f_n(X_{P_0}, \ldots, X_{P_{n-2}}, U) = 0.$$

Therefore, candidates for x-coordinates of the P_i can be found by solving

$$f_n(x_{P_0}, \ldots, x_{P_{n-2}}, U) = 0 \tag{7.4}$$

for the x_{P_i}. We Weil restrict equation (7.4) using the coordinates

$$x_{P_i} = x_{i,0} + x_{i,1}\zeta + \ldots + x_{i,n-1}\zeta^{n-1}$$

and obtain n equations

$$F_j(x_{0,0}, \ldots, x_{n-2,n-1}, U_0, \ldots, U_{n-1}) = 0, \quad j = 0, \ldots, n-1. \tag{7.5}$$

Solving this system over $\mathbb{F}_q$ is equivalent to solving equation (7.4) over $\mathbb{F}_{q^n}$ and yields possible x-coordinates for the points P_i.

In addition to requiring that the P_i sum to R, we must ensure that they are in the factor base. Therefore, we set $x_{i,0} = \ldots = x_{i,n-2} = 0$ for $i = 0, \ldots, n-1$, and we include an equation of the form (7.2) in system (7.5) for each P_i. This means that in order to find a relation, we have to solve

$$\begin{array}{rcl} F_0(0,\dots,0,x_{0,n-2},x_{0,n-1},\dots,0,\dots,0,x_{n-2,n-2},x_{n-2,n-1},U_0,\dots,U_{n-1}) & = & 0 \\ & \vdots & \\ F_{n-1}(0,\dots,0,x_{0,n-2},x_{0,n-1},\dots,0,\dots,0,x_{n-2,n-2},x_{n-2,n-1},U_0,\dots,U_{n-1}) & = & 0 \\ \tilde{f}_n(0,\dots,0,x_{0,n-2},x_{0,n-1}) & = & 0 \\ & \vdots & \\ \tilde{f}_n(0,\dots,0,x_{n-2,n-2},x_{n-2,n-1}) & = & 0 \end{array} \tag{7.6}$$

over $\mathbb{F}_q$. The system has $2n-1$ equations in $2(n-1)$ indeterminates, two indeterminates for each of the P_i. The first n equations are the Weil descent of the n-th Semaev polynomial, where a constant has been plugged in for the last indeterminate. Therefore, they each have total degree $(n-1)2^{n-2}$. They describe that the points P_i, whose x-coordinates are described by the solutions of the system, sum to R, whose x-coordinate is U. The last $n-1$ equations also have total degree $(n-1)2^{n-2}$. They guarantee that the solution points P_i are in the factor base.

Since the system has more equations than unknowns, it is generically of dimension 0, i.e. it has a finite number of solutions over $\overline{\mathbb{F}}_q$ (and it is actually of dimension 0 in all our experiments). Thus, using the Shape Lemma (cf. Theorem 2.30), the system may be solved by computing a lexicographic Gröbner basis and then factoring a univariate polynomial.

Whenever a given point R decomposes over the factor base, i.e. when a relation of the form (7.3) exists, this gives a solution of system (7.6). The converse, however, is not true. For example, when the solutions of the system give x-coordinates where one of the corresponding y-coordinates is not in $\mathbb{F}_{q^n}$, then this does not produce a valid relation.

Remark 7.4. Joux and Vitse [JV12] propose considering relations that involve one factor base point less than suggested by Gaudry, i.e. only $n-2$ points P_i in our case. This reduces the probability of finding relations by a factor q, but in some cases it can make the difference between a manageable and an unmanageable system. We consider this idea in Section 7.3.2.

Finally, we need to produce more relations than there are factor base elements, i.e. about q, by solving the system sufficiently many times (see Section 7.2 for an estimate) for different random points R, keeping in mind that often the system will have no solution or a solution that does not produce a valid relation.

7.1.4 Linear algebra

The relation collection phase of the algorithm produces a sparse matrix of size about $q \times q$ with entries 0 or 1. The rows correspond to the factor base elements, and the columns correspond to the different relations involving a point R, each given by the values α and β. Each column has $n-1$ non-zero entries, one for each factor base element that appears in the corresponding relation. Assuming that more relations have been produced than there are factor base elements, the matrix has more columns than rows. Therefore, there exists a non-zero vector in its right kernel. The task of the linear algebra step is to find such a vector, where the computations must be performed not over $\mathbb{Z}$, but modulo the order of P in T_n. Standard methods to solve such sparse linear systems are Wiedemann's Algorithm and Lanczos' Algorithm (see [Wie86, LO90]).

Remark 7.5. Since there are efficient and well-studied methods for solving sparse linear systems, we do not treat them in detail here. Notice however that the efficient implementation of the linear algebra step is far from trivial, especially since the algorithms are hard to parallelize. One recent record-breaking implementation on GPUs is presented in [Jel13, Jel14]. Moreover, in practice a filtering step can make a big difference, see e.g. [Bou12]. This is a preprocessing of the matrix, where duplicate relations are removed, points (corresponding to rows) that appear in only one relation are removed, and excess relations are removed until there are exactly $|\mathcal{F}|+1$ of them left. We do not employ such sophisticated techniques in our experiments, since we treat only small examples and our emphasis is on finding relations and not on the linear algebra step.

7.1.5 Individual logarithm

Once the linear system has been solved, computing the actual discrete logarithm is easy. Denoting by $\gamma_1, \ldots, \gamma_r$ the entries of the vector in the kernel of the matrix computed in the previous step and by α_j, β_j the values of α, β corresponding to the j-th relation we have

$$\log_P Q = -\left(\sum_{j=1}^r \gamma_j \alpha_j\right)\left(\sum_{j=1}^r \gamma_j \beta_j\right)^{-1},$$

provided that $\sum \gamma_j \beta_j$ is invertible modulo the order or P. If not, one must collect more relations in order to produce a different matrix and find a different vector γ. Notice, however, that $\sum \gamma_j \beta_j$ is invertible with high probability, especially if $\operatorname{ord}(P)$ is prime.

Remark 7.6. We point out that this algorithm works for elliptic curves defined over finite fields of any characteristic. Although the original definition of the Semaev polynomial in [Sem04] is only for curves in short Weierstraß form, it may easily be adjusted to fields of characteristic 2 and 3 (see also Remark 3.9).

7.2 Complexity analysis

We now analyze the complexity of the index calculus algorithm presented in the previous section. We make the same heuristic assumptions as Gaudry [Gau09] and other work based on Gaudry's results, e.g. [GV05, JV12]. Our analysis is in q and n and therefore more precise than that of Gaudry, who considers only varieties of constant dimension. By setting n to be a constant in our analysis, one obtains the result of Gaudry. For simplicity, we use $\tilde{O}$-notation, which ignores logarithmic factors in both n and q.

Setup. Diem [Die11] shows that the n-th Semaev polynomial and its Weil restriction can be computed with a randomized algorithm in expected time polynomial in $\tilde{O}(e^{n^2})$.

Remark 7.7. We do not have to compute the full Weil restriction of $f_n(x_i, x_i^q, \ldots, x_i^{q^{n-1}})$ or of $f_n(x_{P_0}, x_{P_1}, \ldots, x_{P_{n-2}}, u)$, since we only need them for the x-coordinates x_{P_i} of points in the factor base. Therefore, when computing the Weil restriction, we may immediately work with the Weil restriction coordinates $x_{P_i} = x_{i,n-2}\zeta^{n-2} + x_{i,n-1}\zeta^{n-1}$. In practice, this is much quicker than first computing the standard Weil restriction and then setting $x_{i0} = \ldots = x_{i,n-3} = 0$. However, we treat u, the x-coordinate of R, as an indeterminate. Then we only have to compute the Weil restriction once to obtain system (7.6), and we may plug the value of the x-coordinate of R into the system directly.

Factor base. In order to enumerate the factor base, we go through all values $X_{n-2} \in \mathbb{F}_q$, compute the solutions of $\tilde{f}_n(0, \ldots, 0, X_{n-2}, x_{n-1}) = 0$ over $\mathbb{F}_q$, and check whether the solution gives a point in T_n. Since the degree of $\tilde{f}_n$ in x_{n-1} is bounded by $(n-1)2^{n-2}$, computing all solutions takes $\tilde{O}((n-1)2^{n-2})$ operations in $\mathbb{F}_q$ (see [GvzG99, Corollary 14.16]). Typically, there are only few solutions. Checking whether the y-coordinate corresponding to $X = X_{n-2}\zeta^{n-2}+X_{n-1}\zeta^{n-1}$ is in $\mathbb{F}_{q^n}$ is much cheaper. Altogether, enumerating the factor base costs

$$\tilde{O}(q(n-1)2^{n-2}).$$

Relation generation. Let us assume, like Gaudry does, that *different* unordered $(n-1)$-tuples of factor base elements sum to *different* points in T_n most of the time. Then $|\mathcal{F}|^{n-1}/(n-1)!$ points of T_n decompose over the factor base, and since T_n has about q^{n-1} elements, this means that the probability of a point $R \in T_n$ splitting over the factor base is $1/(n-1)!$. Therefore, in order to generate q relations, we expect to have to try to decompose $q(n-1)!$ points, i.e. solve $q(n-1)!$ systems.

In order to solve each system, we follow the approach that is most efficient in practice: We first compute a Gröbner basis of the system with respect to the degree reverse lexicographic term order, and we then use a Gröbner walk algorithm to convert it to a lexicographic Gröbner basis. Afterwards, we factor a univariate polynomial. The last step is negligible compared to the first two.

To estimate the complexity of the Gröbner basis computation, we use the bound on the complexity of Faugère's F5 algorithm [Fau02]. We assume that the system is semi-regular, which is true generically. Then according to [BFSY05, Proposition 6], the complexity of computing a degree reverse lexicographic Gröbner basis of our system, which is zero-dimensional and in $2n-2$ variables, is

$$O\left(\binom{d_{\text{reg}} + 2n - 2}{2n-2}^{\omega}\right),$$

where $2 \leq \omega \leq 3$ is the linear algebra constant (i.e. the exponent in the complexity of matrix multiplication) and d_{reg} is the degree of regularity of the system (this is also called the regularity index, see [KR05, Definition 5.1.8]).

We estimate d_{reg} using a standard bound from commutative algebra (see (2.3) or e.g. [Laz83], this is called *Macaulay bound* in [BFSY05]):

$$d_{\text{reg}} \leq (2n-1)((n-1)2^{n-2} - 1) + 1 = (2n-1)(n-1)2^{n-2} - 2n + 2.$$

Hence the complexity of computing a degree reverse lexicographic Gröbner basis of our system is bounded by

$$O\left(\binom{(2n-1)(n-1)2^{n-2}}{2n-2}^{\omega}\right).$$

Now using the FGLM algorithm [FGLM93], we may compute from this basis a lexicographic Gröbner basis in

$$O((2n-2) \cdot D^3),$$

where D is the degree of the ideal generated by the degree reverse lexicographic Gröbner basis (i.e. the number of solutions counted with multiplicity in $\overline{\mathbb{F}}_q$). Using as a bound on D the product of the degrees of the equations of the system, this is not more expensive than F5.

Taking into account that we have to do this $q(n-1)!$ times, the total cost of the relation collection step is

$$O\left(\binom{(2n-1)(n-1)2^{n-2}}{2n-2}^{\omega}(n-1)!q\right).$$

Linear algebra. Using Lanczos' or Wiedemann's Algorithm, the cost of solving a sparse linear system of size about $q \times q$, where each column has $n-1$ non-zero entries, is

$$O((n-1)q^2)$$

(see e.g. [EK97]).

Individual logarithm. The cost of computing the individual logarithm is clearly negligible compared to the complexities above.

Putting everything together, we get that the algorithm has a total complexity of

$$\tilde{O}\left(\binom{(2n-1)(n-1)2^{n-2}}{2n-2}^{\omega}(n-1)!q^2\right).$$

Double large prime variation. As suggested by Gaudry, we may use the double large prime variation [Thé03, GTTD07] in order to rebalance the complexity of the relation collection and the linear algebra step in q. Then one must collect $q^{2-2/(n-1)}$ relations instead of q and solve a linear system of size $q^{1-1/(n-1)} \times q^{1-1/(n-1)}$ instead of $q \times q$. Then the overall cost of the algorithm becomes

$$\tilde{O}\left(\binom{(2n-1)(n-1)2^{n-2}}{2n-2}^{\omega}(n-1)!q^{2-2/(n-1)}\right).$$

Hence we have proven the following heuristic result.

Theorem 7.8. *Let $T_n, n \geq 3$, be the trace zero subgroup of an elliptic curve. Then there exists a probabilistic algorithm that computes discrete logarithms in T_n in heuristic time*

$$\tilde{O}\left(\binom{(2n-1)(n-1)2^{n-2}}{2n-2}^{\omega}(n-1)!q^{2-2/(n-1)}\right)$$

when q and n tend to infinity.

When considering n to be constant, this gives precisely the heuristic complexity of $\tilde{O}(q^{2-2/(n-1)})$ from [Gau09]. Our analysis makes explicit the exponential dependency on n already pointed out by Gaudry. This is due to the cost of the Gröbner basis computation and cannot be improved for generic systems (it is a well-known fact that in the worst case, Gröbner basis computations are doubly exponential in the degrees of the input equations [MM82]).

7.3 Explicit equations and experiments

We now study the systems of polynomial equations that describe the relations and the overall behavior of our algorithm for $n = 3, 5$. All computations were done with Magma version 2.19.3 [BCP97] on one core of an Intel Xeon X7550 Processor (2.00 GHz) on a Fujitsu Primergy RX900S1. As in the other chapters, we stress that all our implementations are only meant to be a proof of concept. In this section, we use a straightforward implementation of the algorithm described in Section 7.1, and we use the built-in Magma routines wherever possible, e.g. for Gröbner basis computation, polynomial factorization, and linear algebra. Our timings are only meant as an indication, and they could be improved significantly by a special-purpose implementation, using current state-of-the-art methods from index calculus research such as [BBD+14], and by choosing convenient parameters, such as finite fields where particularly efficient arithmetic is possible. We concentrate mostly on the computation of the equations of the trace zero subgroup, the factor base, and the relation generation. In particular, we did not implement any filtering (preprocessing of the matrix, see Remark 7.5), except that we do not allow duplicate relations, we did not implement the double large prime variation, and our implementation is not parallelized.

7.3.1 Explicit equations for $n = 3$

When $n = 3$, the trace zero variety has dimension 2. Therefore, the index calculus attack on T_3 is not better than generic (square root) attacks on T_3. Since $n = 3$ is the case where all equations are small enough to be written down explicitly, we present them nevertheless, together with some experimental data that allows us to make predictions on the feasibility of this attack for different values of q.

As in previous chapters, *we assume that* $3 \mid q-1$ *and write* $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - \mu)$ *as a Kummer extension of* $\mathbb{F}_q$ *with basis* $1, \zeta, \zeta^2$. For cases where this is not possible, see Remark 3.1. *We also assume that* $\mathbb{F}_q$ *does not have characteristic 2 or 3 and that* E *is given by an equation in short Weierstraß form*

$$E : y^2 = x^3 + Ax + B.$$

Our approach also works when $\mathbb{F}_q$ has characteristic 2 or 3, but in this case the definition of the Semaev polynomial and all equations given below must be adjusted accordingly (see Remark 7.6).

The third Semaev polynomial is

$$f_3(z_1, z_2, z_3) = (z_1 - z_2)^2 z_3^2 - 2((z_1 + z_2)(z_1 z_2 + A) + 2B)z_3 + (z_1 z_2 - A)^2 - 4B(z_1 + z_2),$$

and the Weil restriction of $f_3(x, x^q, x^{q^2})$ is

$$\begin{aligned} \tilde{f}_3(x_0, x_1, x_2) \;=\; & -3x_0^4 - 12\mu^2 x_0 x_2^3 - 12\mu x_0 x_1^3 + 18\mu x_0^2 x_1 x_2 \\ & +9\mu^2 x_1^2 x_2^2 - 6Ax_0^2 + 6A\mu x_1 x_2 - 12Bx_0 + A^2, \end{aligned} \tag{7.7}$$

see also equation (3.9). We write the points of T_3 as tuples (X_0, X_1, X_2) that satisfy $\tilde{f}_3(X_0, X_1, X_2) = 0$. For the factor base, we choose those points with $X_0 = 0$, namely

$$\mathcal{F} = \{(0, X_1, X_2) \in T_3\}.$$

These are precisely the points in T_n that satisfy

$$\tilde{f}_3(0, X_1, X_2) = 9\mu^2 X_1^2 X_2^2 + 6A\mu X_1 X_2 + A^2 = (3\mu X_1 X_2 + A)^2 = 0. \tag{7.8}$$

If $A = 0$, then this is equivalent to

$$X_1 X_2 = 0,$$

and it is particularly easy to enumerate the factor base: One simply checks which x-coordinates $(0, X_1, 0)$ and $(0, 0, X_2)$, for $X_1, X_2 \in \mathbb{F}_q$, give points in T_3. If, on the other hand, $A \neq 0$, then every solution of (7.8) satisfies $X_1 \neq 0$, and moreover (7.8) is equivalent to

$$X_2 = -\frac{A}{3\mu X_1}. \tag{7.9}$$

In this case, it is also fairly easy to enumerate the factor base: For every $X_1 \in \mathbb{F}_q^\times$, one computes X_2 according to (7.9) and checks whether this yields a point of T_3.

Now we need to find relations of the form

$$R = P_0 \oplus P_1,$$

where R with x-coordinate $U = U_0 + U_1\zeta + U_2\zeta^2$ is given and P_1, P_2 are in $\mathcal{F}$. We denote by $x_{P_0} = x_{01}\zeta + x_{02}\zeta^2$ the indeterminates representing the x-coordinate of P_0 and by $x_{P_1} = x_{11}\zeta + x_{12}\zeta^2$ those representing the x-coordinate of P_1. Then we have to solve

$$f_3(x_{P_0}, x_{P_1}, U) = 0,$$

or equivalently, its Weil restriction

$$\begin{array}{rcl}
0 & = & -4BU_0 + A^2 + 4\mu^2 x_{01}x_{11}x_{02}x_{12} - 4\mu^2 x_{01}x_{02}x_{11}U_2 - 4\mu^2 x_{01}x_{02}x_{12}U_1 \\
 & & -4\mu^2 x_{11}x_{01}x_{12}U_2 - 4\mu^2 x_{11}x_{02}x_{12}U_1 - 4\mu^2 x_{02}x_{11}U_1U_2 + 4\mu^2 x_{11}x_{12}U_1U_2 \\
 & & -4\mu^2 x_{02}x_{12}U_0U_2 + 4\mu^2 x_{01}x_{02}U_1U_2 - 4\mu^2 x_{01}x_{12}U_1U_2 + 2\mu^2 {x_{02}}^2 U_0U_2 \\
 & & +2\mu^2 {x_{12}}^2 U_0U_2 - 2\mu^2 x_{02}x_{12}{U_1}^2 - 2\mu^2 x_{01}x_{11}{U_2}^2 - 2\mu^2 {x_{02}}^2 x_{11}U_1 \\
 & & -2\mu^2 {x_{02}}^2 x_{12}U_0 - 2\mu^2 {x_{11}}^2 x_{02}U_2 - 2\mu^2 {x_{12}}^2 x_{01}U_1 - 2\mu^2 {x_{12}}^2 x_{02}U_0 \\
 & & -2\mu^2 {x_{01}}^2 x_{12}U_2 - 2\mu x_{02}AU_1 - 2\mu {x_{11}}^2 x_{01}u_0 - 2\mu x_{11}AU_2 - 2\mu x_{12}AU_1 \\
 & & -2\mu x_{01}AU_2 + 2\mu {x_{01}}^2 U_0U_1 + 2\mu {x_{11}}^2 U_0U_1 + 2\mu x_{01}x_{02}{U_0}^2 - 2\mu x_{01}x_{12}{U_0}^2 \\
 & & -2\mu x_{02}x_{11}{U_0}^2 + 2\mu x_{11}x_{12}{U_0}^2 - 2\mu {x_{01}}^2 x_{11}U_0 - 2\mu x_{01}x_{12}A - 2\mu x_{02}x_{11}A \\
 & & +\mu^2 {x_{11}}^2 {U_2}^2 + \mu^2 {x_{02}}^2 {U_1}^2 + \mu^2 {x_{01}}^2 {x_{12}}^2 + \mu^2 {x_{02}}^2 {x_{11}}^2 + \mu^2 {x_{01}}^2 {U_2}^2 \\
 & & +\mu^2 {x_{12}}^2 {U_1}^2 - 4\mu x_{01}x_{11}U_0U_1 \\
0 & = & -4BU_1 - 2x_{11}AU_0 - 2x_{01}AU_0 - 4Bx_{01} - 4Bx_{11} - 4\mu^2 x_{01}x_{02}x_{12}U_2 \\
 & & -4\mu^2 x_{11}x_{02}x_{12}U_2 - 4\mu^2 x_{02}x_{12}U_1U_2 - 4\mu x_{01}x_{02}x_{11}U_0 - 4\mu x_{11}x_{01}x_{12}U_0 \\
 & & -4\mu x_{02}x_{11}U_0U_1 + 4\mu x_{11}x_{12}U_0U_1 - 4\mu x_{01}x_{11}U_0U_2 + 4\mu x_{01}x_{02}U_0U_1 \\
 & & -4\mu x_{01}x_{12}U_0U_1 + 2\mu {x_{01}}^2 U_0U_2 + 2\mu {x_{11}}^2 U_0U_2 - 2\mu x_{02}x_{12}{U_0}^2 - 2\mu x_{01}x_{11}{U_1}^2 \\
 & & -2\mu {x_{01}}^2 x_{11}U_1 - 2\mu x_{02}x_{12}A - 2\mu x_{02}AU_2 - 2\mu {x_{11}}^2 x_{01}U_1 - 2\mu {x_{11}}^2 x_{02}U_0 \\
 & & -2\mu x_{12}AU_2 - 2\mu {x_{01}}^2 x_{12}U_0 + 2\mu^2 {x_{02}}^2 U_1U_2 + 2\mu^2 {x_{12}}^2 U_1U_2 + 2\mu^2 x_{01}x_{02}{U_2}^2 \\
 & & -2\mu^2 x_{01}x_{12}{U_2}^2 - 2\mu^2 x_{02}x_{11}{U_2}^2 + 2\mu^2 x_{11}x_{12}{U_2}^2 - 2\mu^2 {x_{02}}^2 x_{11}U_2 \\
 & & +2\mu^2 x_{01}{x_{12}}^2 x_{02} + 2\mu^2 {x_{02}}^2 x_{11}x_{12} - 2\mu^2 {x_{02}}^2 x_{12}U_1 - 2\mu^2 {x_{12}}^2 x_{01}U_2 \\
 & & -2\mu^2 {x_{12}}^2 x_{02}U_1 + \mu {x_{01}}^2 {U_1}^2 + \mu {x_{01}}^2 {x_{11}}^2 + \mu {x_{11}}^2 {U_1}^2 + \mu {x_{12}}^2 {U_0}^2 + \mu {x_{02}}^2 {U_0}^2 \\
0 & = & -4BU_2 - 2x_{01}x_{11}A - 2x_{02}AU_0 - 2x_{11}AU_1 - 2x_{12}AU_0 - 2x_{01}AU_1 \\
 & & -2x_{01}x_{11}{U_0}^2 + {x_{01}}^2 {U_0}^2 + {x_{11}}^2 {U_0}^2 - 4Bx_{02} - 4Bx_{12} + 4\mu x_{11}x_{12}U_0U_2 \\
 & & -4\mu x_{02}x_{12}U_0U_1 - 4\mu x_{01}x_{11}u_1U_2 + 4\mu x_{01}x_{02}U_0U_2 - 4\mu x_{01}x_{12}U_0U_2 \\
 & & -4\mu x_{01}x_{02}x_{11}U_1 - 4\mu x_{01}x_{02}x_{12}U_0 - 4\mu x_{11}x_{01}x_{12}U_1 - 4\mu x_{11}x_{02}x_{12}U_0 \\
 & & -4\mu x_{02}x_{11}U_0U_2 - 2\mu x_{02}x_{11}{U_1}^2 + 2\mu x_{11}x_{12}{U_1}^2 - 2\mu {x_{01}}^2 x_{11}U_2 \\
 & & -2\mu {x_{02}}^2 x_{11}U_0 + 2\mu {x_{01}}^2 x_{11}x_{12} + 2\mu x_{01}{x_{11}}^2 x_{02} - 2\mu {x_{11}}^2 x_{01}U_2 \\
 & & -2\mu {x_{11}}^2 x_{02}U_1 - 2\mu {x_{12}}^2 x_{01}U_0 - 2\mu {x_{01}}^2 x_{12}U_1 - 2\mu^2 x_{02}x_{12}{U_2}^2 \\
 & & -2\mu^2 {x_{02}}^2 x_{12}U_2 - 2\mu^2 {x_{12}}^2 x_{02}U_2 + 2\mu {x_{01}}^2 U_1U_2 + 2\mu {x_{02}}^2 U_0U_1 \\
 & & +2\mu {x_{11}}^2 U_1U_2 + 2\mu {x_{12}}^2 U_0U_1 + 2\mu x_{01}x_{02}{U_1}^2 - 2\mu x_{01}x_{12}{U_1}^2 \\
 & & +\mu^2 {x_{12}}^2 {U_2}^2 + \mu^2 {x_{02}}^2 {x_{12}}^2 + \mu^2 {x_{02}}^2 {U_2}^2.
\end{array}$$

Assuming that $A \neq 0$, which is the general case, from (7.9) we get

$$x_{02} = -\frac{A}{3\mu x_{01}} \quad \text{and} \quad x_{12} = -\frac{A}{3\mu x_{11}}.$$

Plugging this into the above system and multiplying the first two equations by $27\mu x_{01}^2 x_{11}^2$ and the third equation by $81\mu^2 x_{01}^2 x_{11}^2$ allows us to eliminate the two indeterminates x_{02}

and x_{12} from the system that describes a relation. We obtain

$$\begin{array}{rcl}
0 & = & 36{x_{01}}^3x_{11}\mu^2AU_1U_2 + 36x_{01}{x_{11}}^3\mu^2AU_1U_2 - 72{x_{01}}^2{x_{11}}^2\mu^2AU_1U_2 \\
& & -12x_{01}x_{11}A^2U_0U_2\mu + 54{x_{01}}^4{x_{11}}^2\mu^2U_0U_1 + 54{x_{01}}^2{x_{11}}^4\mu^2U_0U_1 \\
& & -18{x_{01}}^3{x_{11}}^2\mu^2AU_2 - 18{x_{01}}^2{x_{11}}^3\mu^2AU_2 - 108{x_{01}}^3{x_{11}}^3\mu^2U_0U_1 \\
& & +18x_{01}{x_{11}}^4\mu^2AU_2 + 18{x_{01}}^4x_{11}\mu^2AU_2 + 6{x_{01}}^2A^2U_0U_2\mu + 6{x_{11}}^2A^2U_0U_2\mu \\
& & -108{x_{01}}^2{x_{11}}^2BU_0\mu - 36{x_{01}}^2{x_{11}}^2A{U_0}^2\mu + 6{x_{01}}^2x_{11}A^2U_1\mu + 6x_{01}{x_{11}}^2A^2U_1\mu \\
& & +18{x_{01}}^3x_{11}A{U_0}^2\mu - 6x_{01}x_{11}A^2{U_1}^2\mu + 18x_{01}{x_{11}}^3A{U_0}^2\mu + 3{x_{11}}^4A^2\mu \\
& & +3{x_{01}}^4A^2\mu - 54{x_{01}}^4{x_{11}}^3\mu^2U_0 - 54{x_{01}}^3{x_{11}}^4\mu^2U_0 - 54{x_{01}}^3{x_{11}}^3\mu^3{U_2}^2 \\
& & +27{x_{01}}^2{x_{11}}^4\mu^3{U_2}^2 + 27{x_{01}}^4{x_{11}}^2\mu^3{U_2}^2 + 18x_{01}{x_{11}}^3A^2\mu + 18{x_{01}}^3x_{11}A^2\mu \\
& & +39{x_{01}}^2{x_{11}}^2A^2\mu + 3{x_{11}}^2A^2{U_1}^2\mu + 3{x_{01}}^2A^2{U_1}^2\mu - 6{x_{01}}^3A^2U_1\mu \\
& & -6{x_{11}}^3A^2U_1\mu + 2x_{01}A^3U_0 + 2x_{11}A^3U_0 \\
0 & = & -72{x_{01}}^2{x_{11}}^2AU_0U_1\mu + 36x_{01}{x_{11}}^3AU_0U_1\mu - 12x_{01}x_{11}A^2U_1U_2\mu \\
& & +36{x_{01}}^3x_{11}AU_0U_1\mu - 6x_{01}x_{11}A^3 + 3{x_{01}}^2A^2{U_0}^2 + 2x_{11}A^3U_1 + 2x_{01}A^3U_1 \\
& & +3{x_{11}}^2A^2{U_0}^2 - 54{x_{01}}^3{x_{11}}^4\mu^2U_1 + 27{x_{01}}^2{x_{11}}^4\mu^2{U_1}^2 + 27{x_{01}}^4{x_{11}}^2\mu^2{U_1}^2 \\
& & -54{x_{01}}^3{x_{11}}^3\mu^2{U_1}^2 - 54{x_{01}}^4{x_{11}}^3\mu^2U_1 - 6{x_{11}}^3A^2U_2\mu - 6{x_{01}}^3A^2U_2\mu \\
& & -108{x_{01}}^2{x_{11}}^3B\mu - 108{x_{01}}^3{x_{11}}^2B\mu - 2{x_{11}}^2A^3 + 27{x_{01}}^4{x_{11}}^4\mu^2 - 2{x_{01}}^2A^3 \\
& & -6x_{01}x_{11}A^2{U_0}^2 + 54{x_{01}}^2{x_{11}}^4\mu^2U_0U_2 + 54{x_{01}}^4{x_{11}}^2\mu^2U_0U_2 \\
& & -108{x_{01}}^3{x_{11}}^3\mu^2U_0U_2 - 36{x_{01}}^2{x_{11}}^2\mu^2A{U_2}^2 + 18x_{01}{x_{11}}^3\mu^2A{U_2}^2 \\
& & +18{x_{01}}^3x_{11}\mu^2A{U_2}^2 - 18{x_{01}}^3{x_{11}}^2AU_0\mu - 18{x_{01}}^2{x_{11}}^3AU_0\mu + 6x_{01}{x_{11}}^2A^2U_2\mu \\
& & +6{x_{01}}^2x_{11}A^2U_2\mu - 108{x_{01}}^2{x_{11}}^2BU_1\mu + 18x_{01}{x_{11}}^4AU_0\mu + 18{x_{01}}^4x_{11}AU_0\mu \\
& & +6{x_{01}}^2A^2U_1U_2\mu + 6{x_{11}}^2A^2U_1U_2\mu \\
0 & = & -216{x_{01}}^2{x_{11}}^2AU_0U_2\mu^2 + 108x_{01}{x_{11}}^3AU_0U_2\mu^2 + 108{x_{01}}^3x_{11}AU_0U_2\mu^2 \\
& & +18{x_{11}}^2A^2U_0U_1\mu + 18{x_{01}}^2A^2U_0U_1\mu + 18{x_{01}}^2x_{11}A^2U_0\mu + 108x_{01}{x_{11}}^2BA\mu \\
& & +18x_{01}{x_{11}}^2A^2U_0\mu + 108{x_{01}}^2x_{11}BA\mu - 36x_{01}x_{11}A^2U_0U_1\mu - 162{x_{01}}^4{x_{11}}^3\mu^3U_2 \\
& & -162{x_{01}}^3{x_{11}}^4\mu^3U_2 + 9{x_{11}}^2A^2{U_2}^2\mu^2 + 9{x_{01}}^2A^2{U_2}^2\mu^2 - 162{x_{01}}^3{x_{11}}^3A\mu^2 \\
& & +81{x_{01}}^2{x_{11}}^4{U_0}^2\mu^2 - 54{x_{01}}^2{x_{11}}^4A\mu^2 + 81{x_{01}}^4{x_{11}}^2{U_0}^2\mu^2 - 162{x_{01}}^3{x_{11}}^3{U_0}^2\mu^2 \\
& & -54{x_{01}}^4{x_{11}}^2A\mu^2 + A^4 - 324{x_{01}}^2{x_{11}}^2BU_2\mu^2 + 54{x_{01}}^4x_{11}AU_1\mu^2 \\
& & +54{x_{01}}^3x_{11}A{U_1}^2\mu^2 - 18x_{01}x_{11}A^2{U_2}^2\mu^2 + 54x_{01}{x_{11}}^4AU_1\mu^2 + 54x_{01}{x_{11}}^3A{U_1}^2\mu^2 \\
& & -54{x_{01}}^3{x_{11}}^2AU_1\mu^2 - 54{x_{01}}^2{x_{11}}^3AU_1\mu^2 - 108{x_{01}}^2{x_{11}}^2A{U_1}^2\mu^2 \\
& & +162{x_{01}}^4{x_{11}}^2\mu^3U_1U_2 - 324{x_{01}}^3{x_{11}}^3\mu^3U_1U_2 + 162{x_{01}}^2{x_{11}}^4\mu^3U_1U_2 \\
& & -18{x_{11}}^3A^2U_0\mu + 6x_{11}A^3U_2\mu - 18{x_{01}}^3A^2U_0\mu + 6x_{01}A^3U_2\mu.
\end{array} \tag{7.10}$$

This system is defined only in the two indeterminates x_{01}, x_{11}. All equations have degree 4 in both x_{01} and x_{11}. The first and third equations have total degree 7, and the second equation has total degree 8. We have computed that the system (7.10) has regularity 14 for almost all points R (and regularity 12 or 13 for some special choices of R). This means that the highest degree of all polynomials appearing during the Gröbner basis computation is at most 14. This moderate number suggests that the Gröbner basis computation is not very costly, and our experiments (see below) show that this is indeed true.

For a given x-coordinate U of a point $R \in T_n$, the $\mathbb{F}_q$-solutions (X_{01}, X_{11}) of the above system with $X_{01}, X_{11} \neq 0$ give candidates for x-coordinates

$$X_0 = X_{01}\zeta - \frac{A}{3\mu X_{01}}\zeta^2 \quad \text{and} \quad X_1 = X_{11}\zeta - \frac{A}{3\mu X_{11}}\zeta^2$$

of the points P_0, P_1 in the relation.

Example 7.9. We give a toy example. Let $q = 2^{12} - 3, \mathbb{F}_{q^3} = \mathbb{F}_q/(\zeta^3 - 2)$, and $E : y^2 = x^3 + x + 21$. Then T_3 has order 16715869, which is a 24-bit prime, and we take

$$P = 3961 + 199\zeta + 4028\zeta^2$$

as a generator. We choose a random

$$Q = 3342 + 3020\zeta + 4031\zeta^2,$$

of which we wish to compute the discrete logarithm. The elements of the factor base satisfy

$$6x_{i1}x_{i2} + 1 = 0, \quad i = 0, 1,$$

and we compute that there are exactly 4002 such points. Now we choose random $\alpha = 4297188$ and $\beta = 10382682$, which gives $U = 2960 + 1129\zeta + 1917\zeta^2$, and we solve the system

$$\begin{aligned}
0 &= 439x_{01}^4x_{11}^3 + 1215x_{01}^4x_{11}^2 + 2556x_{01}^4x_{11} + 2274x_{01}^4 + 439x_{01}^3x_{11}^4 + 1663x_{01}^3x_{11}^3 \\
&\quad +1537x_{01}^3x_{11}^2 + 3403x_{01}^3x_{11} + 2023x_{01}^3 + 1215x_{01}^2x_{11}^4 + 1537x_{01}^2x_{11}^3 + 1961x_{01}^2x_{11}^2 \\
&\quad +2070x_{01}^2x_{11} + 2326x_{01}^2 + 2556x_{01}x_{11}^4 + 3403x_{01}x_{11}^3 + 2070x_{01}x_{11}^2 + 3534x_{01}x_{11} \\
&\quad +716x_{01} + 2274x_{11}^4 + 2023x_{11}^3 + 2326x_{11}^2 + 716x_{11} \\
0 &= 2x_{01}^4x_{11}^4 + 3670x_{01}^4x_{11}^3 + 938x_{01}^4x_{11}^2 + 609x_{01}^4x_{11} + 3670x_{01}^3x_{11}^4 + 2217x_{01}^3x_{11}^3 \\
&\quad +3400x_{01}^3x_{11}^2 + 405x_{01}^3x_{11} + 3667x_{01}^3 + 938x_{01}^2x_{11}^4 + 3400x_{01}^2x_{11}^3 + 2586x_{01}^2x_{11}^2 \\
&\quad +426x_{01}^2x_{11} + 94x_{01}^2 + 609x_{01}x_{11}^4 + 405x_{01}x_{11}^3 + 426x_{01}x_{11}^2 + 115x_{01}x_{11} \\
&\quad +345x_{01} + 3667x_{11}^3 + 94x_{11}^2 + 345x_{11} \\
0 &= 518x_{01}^4x_{11}^3 + 1692x_{01}^4x_{11}^2 + 2117x_{01}^4x_{11} + 518x_{01}^3x_{11}^4 + 2070x_{01}^3x_{11}^3 + 1976x_{01}^3x_{11}^2 \\
&\quad +1677x_{01}^3x_{11} + 1945x_{01}^3 + 1692x_{01}^2x_{11}^4 + 1976x_{01}^2x_{11}^3 + 3431x_{01}^2x_{11}^2 + 2162x_{01}^2x_{11} \\
&\quad +1057x_{01}^2 + 2117x_{01}x_{11}^4 + 1677x_{01}x_{11}^3 + 2162x_{01}x_{11}^2 + 1979x_{01}x_{11} + 71x_{01} \\
&\quad +1945x_{11}^3 + 1057x_{11}^2 + 71x_{11} + 3474.
\end{aligned}$$

We get $X_{01} = 1770, X_{11} = 1515$, and from these we compute $X_{02} = 338, X_{12} = 3029$, which gives a relation

$$P_0 \oplus P_1 = R$$

for some choice of y-coordinates. After collecting 4002 more such relations and solving the linear system, we obtain $\log_P Q = 419$.

Remark 7.10. During the relation collection phase, a system of similar shape must be solved many times. This is typical for such an index calculus attack, and different techniques exist to make relation generation more efficient. One idea is to compute a parametric Gröbner basis [Wei92, MW10], where $U_0, \dots, U_{n-1}$ are treated as parameters. The idea is to carry out a Gröbner basis computation only once, and afterwards, to immediately obtain a Gröbner basis of of the system where numbers are plugged in for the parameters by plugging the numbers into the parametric Gröbner basis. However, the cost of such computations is very large, and we were not able to compute a parametric Gröbner basis even in this small case where $n = 3$. A more efficient approach, called Gröbner trace algorithm, is due to Traverso [Tra88] and was implemented by Joux and Vitse [JV11]. Their variant of the F4 Gröbner basis algorithm is particularly suitable for computing Gröbner bases of a number of systems of the same shape, since it can reuse information from the first computation for all subsequent ones. However, since we were readily able to solve our system for $n = 3$, and since this is not the interesting case in terms of complexity, we did not try to use their algorithm.

Finally, we present implementation results for fields of different size in Table 7.1. For primes q of $10, 12, 14, 16, 18, 20, 30, 40, 50, 60, 70$, and 80 bits, we chose the smallest possible value μ, and we chose curves E, given by the coefficients A, B, that yield cyclic trace zero subgroups T_3 of prime order. Where we were able to compute it, we list the exact size of the factor base. In all cases, it is close to $q - q^{1/2}$. We also list the number of points R we had to try in order to find $|\mathcal{F}| + 1$ distinct relations.

Times are given in seconds, and numbers in normal font stand for computations that we were able to perform, while numbers in bold represent expected times, extrapolated from timings we were able to obtain. For example, when we are able to compute one relation,

Table 7.1 Index calculus algorithm for $n = 3$, timings in seconds

$\log_2 \lvert T_3 \rvert$	20	24	28	32	36	40
q	$2^{10}-3$	$2^{12}-3$	$2^{14}-3$	$2^{16}-15$	$2^{18}-93$	$2^{20}-3$
μ	5	2	2	2	2	2
A	2	1	1	1	1	1
B	0	21	11	5	10	25
$\lvert\mathcal{F}\rvert$	900	4002	16380	65388	261822	1045962
number of R's tried	2208	8263	32828	130533	522935	2091965
time for GB of large system	0.01773	0.01698	0.01705	0.01792	0.01686	0.01703
time for GB of small system	0.00102	0.00169	0.00167	0.00124	0.00146	0.00135
time to solve small system	0.00115	0.00180	0.00173	0.00134	0.00159	0.00136
time to enumerate $\mathcal{F}$	0.07	0.28	1.15	5.24	23.59	104.86
time to collect relations	3.52	13.53	49.71	197.17	803.95	2845.01
time linear algebra	0.01	0.30	5.22	108.29	129.69	–
total time	3.60	14.25	56.08	310.70	957.23	–
$\log_2 \lvert T_3 \rvert$	60	80	100	120	140	160
q	$2^{30}-105$	$2^{40}-87$	$2^{50}-51$	$2^{60}-93$	$2^{70}-267$	$2^{79}-67$
μ	2	2	2	2	5	3
A	1	1	1	1	1	1
B	24	49	40	193	15	368
$\lvert\mathcal{F}\rvert$	$\mathbf{2^{30}}$	$\mathbf{2^{40}}$	$\mathbf{2^{50}}$	$\mathbf{2^{60}}$	$\mathbf{2^{70}}$	$\mathbf{2^{79}}$
number of R's tried	$\mathbf{2^{31}}$	$\mathbf{2^{41}}$	$\mathbf{2^{51}}$	$\mathbf{2^{61}}$	$\mathbf{2^{71}}$	$\mathbf{2^{80}}$
time for GB of large system	0.02683	0.12645	0.12817	0.13431	0.15000	0.14102
time for GB of small system	0.00146	0.00231	0.00244	0.00249	0.00304	0.00262
time to solve small system	0.00171	0.00291	0.00342	0.00351	0.00467	0.00442
time to enumerate $\mathcal{F}$	$\mathbf{2^{17.2}}$	$\mathbf{2^{28.3}}$	$\mathbf{2^{38.5}}$	$\mathbf{2^{48.7}}$	$\mathbf{2^{59.4}}$	$\mathbf{2^{68.4}}$
time to collect relations	$\mathbf{2^{21.8}}$	$\mathbf{2^{32.5}}$	$\mathbf{2^{42.8}}$	$\mathbf{2^{52.8}}$	$\mathbf{2^{63.2}}$	$\mathbf{2^{72.1}}$

this allows us to predict the time it would take to collect q relations (this requires solving about $2q$ polynomial systems). Where we were not able to carry out a computation or make a prediction, we write "–".

For all field sizes, we were able to solve the system at least a few times. For comparison, we give the time taken to compute a lexicographic Gröbner basis of the straightforward system consisting of 5 equations in 4 indeterminates ("large system"), as well as the time taken to compute a lexicographic Gröbner basis of system (7.10) consisting of 3 equations in 2 indeterminates ("small system"). This shows that this little trick to simplify the system saves a considerable amount of time in practice. Therefore, in the following, we work with the small system.

Next we list in the table the average time taken to solve the small system once. This includes computing the lexicographic Gröbner basis, factoring a univariate polynomial (of degree 6 in our experiments), which gives the value(s) of one indeterminate, and computing the corresponding value(s) of the other indeterminate. For the Gröbner basis computation, we use Magma's `GroebnerBasis()`, which computes a degree reverse lexicographic Gröbner basis using Faugère's F4 algorithm [Fau99] and subsequently a lexicographic Gröbner basis using the FGLM algorithm [FGLM93].

Finally, we give the actual or conjectured times for the full execution of the different steps of our algorithm. First we give the time to enumerate the factor base, then the time to collect $\lvert\mathcal{F}\rvert + 1$ relations, and then the time to solve the linear system, using the sparse linear algebra routine `ModularSolution(Lanczos:=true)` of Magma, which is an implementation of Lanczos' algorithm. We also give the total time to compute one discrete logarithm with our algorithm.

We see that the largest trace zero subgroup where we can compute a full discrete logarithm has 36-bit size. The attack takes approximately 15 minutes. For the 40-bit trace zero subgroup, we can compute sufficiently many relations in about 47 minutes, but we are not able to solve the linear system of size about $2^{20} \times 2^{20}$ in Magma. A specialized implementation presented in [BBD+14, Jel13, Jel14] solves a linear system of size about $2^{22} \times 2^{22}$ in less than 5 days using a sophisticated implementation of Lanczos' algorithm, running on a high performance computer. This means that our attack is certainly feasible for a 40-bit trace zero subgroup. However, we can do much better by rebalancing the cost of relation collection and linear algebra.

Let us consider the group T_3 of 60 bits, with $q \approx 2^{30}$, as given in Table 7.1. We rebalance the complexity with a relatively straightforward approach. Using a factor base of $q^r = 2^{30r}$ elements, where $0 < r < 1$, the probability of finding a relation becomes $q^{2r-2}/2$. Hence in order to find q^r relations, we need to solve $2q^{2-r} = 2^{61-30r}$ systems. Since we know that solving a linear system of size $2^{22} \times 2^{22}$ is possible, we set $q^r = 2^{22}$ and get $r = 0.73$. This means that we would have to collect 2^{39} relations, which would take $2^{39.8}$ seconds or about 30 years. Assuming that solving a linear system of size $2^{23} \times 2^{23}$ is possible, we would need about 15 years to collect relations, etc. We stress that these predictions correspond to the time required by our simple implementation. With an optimized and parallel implementation of the relation collection step (notice that the relation search can trivially be parallelized), it would become faster by a considerable factor. Hence we conclude that with an optimized implementation, computing a discrete logarithm in a 60-bit trace zero subgroup with this index calculus algorithm is feasible.

Interestingly, the crossover point between a Pollard–Rho attack on $E(\mathbb{F}_{q^3})$ and an index calculus attack in T_3 is also at about $q = 2^{30}$. For the curve listed in Table 7.1, Magma can perform about 66000 point additions per second. Since $|E(\mathbb{F}_{q^3})| \approx 2^{90}$, the Pollard–Rho algorithm would have to perform about 2^{45} point additions, which would take about 17 years. However, we computed this number only for interest. If one seriously wanted to attack the DLP in $E(\mathbb{F}_{q^3})$, one should use the Pollard–Rho algorithm in T_3, which would require 2^{30} point additions in T_3. This would take only about 5 minutes.

7.3.2 Explicit equations for $n = 5$

We proceed similarly for $n = 5$, but we cannot write down the equations in this case because they are too large. We assume that $5 \mid q-1$ and write $\mathbb{F}_{q^5} = \mathbb{F}_q(\zeta)/(\zeta^5 - \mu)$. Then $1, \zeta, \zeta^2, \zeta^3, \zeta^4$ is a basis of $\mathbb{F}_{q^5}|\mathbb{F}_q$, which we use for Weil restriction.

The fifth Semaev polynomial f_5 has total degree 32. The same is true for $\tilde{f}(x_0, \ldots, x_4)$ (see also Chapter 3.3.2), which we use as an equation for T_5. The factor base is

$$\mathcal{F} = \{(0, 0, 0, X_3, X_4) \in T_5\},$$

and all its elements satisfy the equation

$$\tilde{f}(0, 0, 0, x_3, x_4) = 0.$$

It has total degree 32 and degree 30 in each x_3 and x_4. Although this polynomial does not have such a simple shape as the corresponding one for $n = 3$, it is still easy to enumerate the factor base: For every $X_3 \in \mathbb{F}_q$, solve $\tilde{f}(0, 0, 0, X_3, x_4) = 0$ for x_4 in $\mathbb{F}_q$.

In order to find relations of the form

$$R = P_0 \oplus P_1 \oplus P_2 \oplus P_3,$$

we compute the Weil descent of $f_5(x_{P_0}, x_{P_1}, x_{P_2}, x_{P_3}, U)$. This gives five equations, each of total degree 32, in 8 indeterminates $x_{i3}, x_{i4}, i = 0, 1, 2, 3$. For a given U, we must solve this system in order to obtain possible relations.

Following the idea of Joux and Vitse [JV12] (see Remark 7.4), where R is decomposed into a sum of 3 factor base elements, we obtain a system of 8 equations in 6 indeterminates, where the first 5 equations (describing that the points sum to R, this is the Weil restriction of $f_4(x_{P_0}, x_{P_1}, x_{P_2}, U)$) have total degree 12 and degree 4 in each indeterminate, and the last 3 equations (describing that the points are in the factor base) are as above. However, this system is too large to be solved with Magma, even over a relatively small field ($\mathbb{F}_{1021}$ in our experiments) and after several weeks of computation and using more than 300 GB of memory.

Since we are not able to solve even one instance of system (7.6) directly, we use a hybrid approach [YCC04, BFP08]. This allows us to find some relations, but it is not fast enough for an attack of realistic cryptographic size. Nevertheless, we give some experimental results, timings, and extrapolations. The hybrid method is often used where a direct Gröbner basis computation is too costly, since it is a trade-off between exhaustive search and Gröbner basis techniques. The main idea is to choose fixed values for a small number of variables and to solve the system in the remaining indeterminates. In order to find all solutions of the system, all possible choices for the fixed variables have to be tried. Therefore, this requires computing many Gröbner bases of smaller systems instead of computing one Gröbner basis of a larger system.

In our case, it is enough to choose one fixed value in order to solve the system readily. We start from the system that describes a relation with 3 factor base elements (i.e. following the approach of [JV12]) and fix $x_{03} = X_{03} \in \mathbb{F}_q$. We then use the factor base equation $\tilde{f}_5(0, 0, 0, X_{03}, x_{04}) = 0$ to determine possible values of x_{04}. Although this equation has degree 30 in x_{04}, there are usually only very few solutions, most frequently 1, 2, or 3. In every case where $x_{04} = X_{04}$ gives a point in the factor base, we plug $x_{03} = X_{03}$ and $x_{04} = X_{04}$ into the system to obtain a new system of 7 equations in the 4 indeterminates $x_{13}, x_{14}, x_{23}, x_{24}$. The first five equations each have total degree 8 and degree 4 in every indeterminate. By trying all $X_{03} \in \mathbb{F}_q$, we find out whether R decomposes over the factor base.

We give some timings and extrapolations in Table 7.2. As before, numbers in normal font are times we measured, and numbers in bold are predictions. After giving the parameters of the fields and curves we used, we indicate the number of points R which we tried to decompose (we expect $6q$), the total number of polynomial systems to be solved for this (we expect $6q^2$), the time for the solution of one system (this is equal to the time for computing a Gröbner basis, since the rest of the computation to solve the system is negligible), the time to enumerate the factor base, the time to collect about q relations, the time for the linear algebra step, and the time for the total attack.

The numbers show that we are able to compute a discrete logarithm in the 27-bit group T_5 in about 2 days and that a discrete logarithm in the 32-bit, 36-bit, and 40-bit groups T_5 can be computed in about 10, 44, and 165 days, respectively. Clearly, this is approach is far from feasible for any group of cryptographic size.

We see that it is very costly to find a relation with this approach, for two reasons. Firstly, we are searching for relations that involve only 3 points of the factor base. While the probability that a point decomposes into a sum of 4 points of the factor base is $1/4! = 1/24$, the probability that it decomposes into a sum of 3 points of the factor base is $1/3!q = 1/6q$ (using similar heuristics as the complexity analysis, see Section 7.2). This means that we expect to have to try about $6q$ points R in order to find one that decomposes. Notice that

Table 7.2 Index calculus algorithm for $n = 5$, timings in seconds

$\log_2 \lvert T_5 \rvert$	20	22	27	32	36	40
q	$2^5 - 1$	$2^6 - 23$	$2^7 - 27$	$2^8 - 15$	$2^9 - 21$	$2^{10} - 3$
μ	2	2	2	3	2	2
A	1	1	1	1	1	1
B	16	3	3	13	18	1
$\lvert\mathcal{F}\rvert$	40	70	110	230	520	970
number of R's tried	886	884	2424	**5784**	**11784**	**24528**
number of systems solved	17719	30934	244824	**1393944**	**5785944**	**25043088**
time for GB of one system	1.30	1.31	1.28	1.21	1.22	1.32
time to enumerate $\mathcal{F}$	0.02	0.04	0.07	0.18	0.43	0.89
time to collect relations	25004	38219	171085	**821328**	**3818016**	**15084720**
time linear algebra	0.01	0.01	0.01	0.01	0.01	0.01
total time	25164	43618	171085	**821328**	**3818016**	**15084720**

$\log_2 \lvert T_5 \rvert$	60	80	100	120	140	160
q	$2^{15} - 157$	$2^{20} - 5$	$2^{25} - 61$	$2^{30} - 173$	$2^{35} - 547$	$2^{40} - 195$
μ	3	2	2	2	5	2
A	1	1	1	1	1	1
B	7	10	17	5	3	12
$\lvert\mathcal{F}\rvert$	32600	1051440	$\mathbf{2^{25}}$	$\mathbf{2^{30}}$	$\mathbf{2^{35}}$	$\mathbf{2^{40}}$
number of R's tried	$\mathbf{2^{20}}$	$\mathbf{2^{25}}$	$\mathbf{2^{30}}$	$\mathbf{2^{35}}$	$\mathbf{2^{40}}$	$\mathbf{2^{45}}$
number of systems solved	$\mathbf{2^{35}}$	$\mathbf{2^{45}}$	$\mathbf{2^{55}}$	$\mathbf{2^{65}}$	$\mathbf{2^{75}}$	$\mathbf{2^{85}}$
time for GB of one system	1.34	1.33	7.09	6.93	146.16	147.89
time to enumerate $\mathcal{F}$	38.80	1530.91	$\mathbf{2^{17.1}}$	$\mathbf{2^{22.9}}$	$\mathbf{2^{28.7}}$	$\mathbf{2^{34.0}}$
time to collect relations	$\mathbf{2^{34.3}}$	$\mathbf{2^{45.4}}$	$\mathbf{2^{57.8}}$	$\mathbf{2^{67.7}}$	$\mathbf{2^{82.2}}$	$\mathbf{2^{92.2}}$
time linear algebra	89.12	–	–	–	–	–
total time	$\mathbf{2^{34.3}}$	$\mathbf{2^{45.4}}$	$\mathbf{2^{57.8}}$	$\mathbf{2^{67.7}}$	$\mathbf{2^{82.2}}$	$\mathbf{2^{92.2}}$

we can hope to find enough relations, even though the probability of finding a relation has decreased by a factor q (Joux and Vitse [JV12] have shown that such an approach is indeed advantageous in certain situations): Assuming that distinct unordered 3-tuples of factor base elements usually sum to distinct points of T_n, this means that about $q^3/6$ points $R \in T_n$ decompose into a sum of 3 factor base elements. This number is much larger than q. Therefore, it is a realistic assumption that we find about q relations.

Secondly, every time we wish to check whether a given point R decomposes into a sum of 3 factor base points, we do not have to solve one system, but $O(q)$ systems, namely a small number of systems for every $X_{03} \in \mathbb{F}_q$. In practice, not all X_{03} yield valid X_{04}, therefore the number of systems to be solved is actually smaller.

Example 7.11. Finally, we give a toy example for our approach. Let $q = 41, \mathbb{F}_{q^5} = \mathbb{F}_q/(\zeta^5 - 2)$, and $E : y^2 = x^3 + x + 3$. Then T_5 has order 2970161, which is a 22-bit prime, and we take

$$P = 5 + 40\zeta + 37\zeta^2 + 38\zeta^3 + 33\zeta^4$$

as a generator. We choose a random

$$Q = 17 + 39\zeta + 34\zeta^2 + 10\zeta^3 + 27\zeta^4,$$

of which we wish to compute the discrete logarithm. The elements of the factor base satisfy

$$\begin{array}{rcl} 0 & = & 37x_{i3}^{30}+13x_{i3}^{28}x_{i4}^{4}+5x_{i3}^{27}x_{i4}+11x_{i3}^{26}x_{i4}^{3}+32x_{i3}^{25}x_{i4}^{5}+19x_{i3}^{25}+21x_{i3}^{24}x_{i4}^{2}+29x_{i3}^{23}x_{i4}^{9} \\ & & +12x_{i3}^{23}x_{i4}^{4}+33x_{i3}^{22}x_{i4}^{6}+5x_{i3}^{22}x_{i4}+6x_{i3}^{21}x_{i4}^{8}+29x_{i3}^{21}x_{i4}^{3}+19x_{i3}^{20}x_{i4}^{10}+33x_{i3}^{20}x_{i4}^{5} \\ & & +24x_{i3}^{20}+30x_{i3}^{19}x_{i4}^{7}+23x_{i3}^{19}x_{i4}^{2}+9x_{i3}^{18}x_{i4}^{14}+35x_{i3}^{18}x_{i4}^{9}+9x_{i3}^{18}x_{i4}^{4}+x_{i3}^{17}x_{i4}^{11} \\ & & +32x_{i3}^{17}x_{i4}^{6}+27x_{i3}^{17}x_{i4}+6x_{i3}^{16}x_{i4}^{13}+31x_{i3}^{16}x_{i4}^{8}+11x_{i3}^{16}x_{i4}^{3}+28x_{i3}^{15}x_{i4}^{10}+40x_{i3}^{15}x_{i4}^{5} \\ & & +26x_{i3}^{15}+33x_{i3}^{14}x_{i4}^{12}+35x_{i3}^{14}x_{i4}^{7}+2x_{i3}^{14}x_{i4}^{2}+14x_{i3}^{13}x_{i4}^{19}+40x_{i3}^{13}x_{i4}^{14}+35x_{i3}^{13}x_{i4}^{9} \\ & & +5x_{i3}^{13}x_{i4}^{4}+2x_{i3}^{12}x_{i4}^{16}+25x_{i3}^{12}x_{i4}^{11}+9x_{i3}^{12}x_{i4}^{6}+4x_{i3}^{12}x_{i4}+34x_{i3}^{11}x_{i4}^{18}+11x_{i3}^{11}x_{i4}^{13} \\ & & +x_{i3}^{11}x_{i4}^{8}+23x_{i3}^{11}x_{i4}^{3}+27x_{i3}^{10}x_{i4}^{20}+21x_{i3}^{10}x_{i4}^{15}+8x_{i3}^{10}x_{i4}^{10}+18x_{i3}^{10}x_{i4}^{5}+38x_{i3}^{10} \\ & & +3x_{i3}^{9}x_{i4}^{17}+6x_{i3}^{9}x_{i4}^{12}+18x_{i3}^{9}x_{i4}^{7}+36x_{i3}^{9}x_{i4}^{2}+37x_{i3}^{8}x_{i4}^{24}+3x_{i3}^{8}x_{i4}^{19}+36x_{i3}^{8}x_{i4}^{14} \\ & & +36x_{i3}^{8}x_{i4}^{9}+33x_{i3}^{8}x_{i4}^{4}+21x_{i3}^{7}x_{i4}^{16}+10x_{i3}^{7}x_{i4}^{11}+38x_{i3}^{7}x_{i4}^{6}+29x_{i3}^{7}x_{i4}+27x_{i3}^{6}x_{i4}^{23} \\ & & +19x_{i3}^{6}x_{i4}^{18}+3x_{i3}^{6}x_{i4}^{13}+11x_{i3}^{6}x_{i4}^{8}+8x_{i3}^{6}x_{i4}^{3}+12x_{i3}^{5}x_{i4}^{25}+24x_{i3}^{5}x_{i4}^{20}+9x_{i3}^{5}x_{i4}^{15} \\ & & +33x_{i3}^{5}x_{i4}^{10}+23x_{i3}^{5}x_{i4}^{5}+7x_{i3}^{5}+11x_{i3}^{4}x_{i4}^{22}+13x_{i3}^{4}x_{i4}^{17}+19x_{i3}^{4}x_{i4}^{12}+26x_{i3}^{4}x_{i4}^{7} \\ & & +35x_{i3}^{4}x_{i4}^{2}+40x_{i3}^{3}x_{i4}^{29}+34x_{i3}^{3}x_{i4}^{24}+29x_{i3}^{3}x_{i4}^{19}+21x_{i3}^{3}x_{i4}^{14}+6x_{i3}^{3}x_{i4}^{9}+2x_{i3}^{3}x_{i4}^{4} \\ & & +31x_{i3}^{2}x_{i4}^{26}+20x_{i3}^{2}x_{i4}^{21}+21x_{i3}^{2}x_{i4}^{16}+17x_{i3}^{2}x_{i4}^{11}+5x_{i3}^{2}x_{i4}^{6}+8x_{i3}^{2}x_{i4}+17x_{i3}x_{i4}^{28} \\ & & +39x_{i3}x_{i4}^{23}+12x_{i3}x_{i4}^{18}+24x_{i3}x_{i4}^{13}+32x_{i3}x_{i4}^{8}+26x_{i3}x_{i4}^{3}+31x_{i4}^{30}+34x_{i4}^{25}+15x_{i4}^{20} \\ & & +3x_{i4}^{15}+29x_{i4}^{10}+14x_{i4}^{5}+10 \end{array}$$

for $i = 0, 1, 2$, and we compute that there are exactly 70 such points. Now we choose randomly $\alpha = 1432283$ and $\beta = 659863$, which gives $U = 3+32\zeta+14\zeta^2+13\zeta^3+37\zeta^4$. We pick $X_{03} = 6$ and solve for $X_{04} = 24$, then the system has the solution $X_{13} = 21, X_{14} = 24, X_{23} = 32, X_{24} = 8$, which gives a relation. After collecting 70 more such relations and solving the linear system, we obtain $\log_P Q = 1041379$.

7.4 Comparison with other attacks and discussion

We now compare the index calculus attack on the DLP in T_n with other known attacks.

Pollard–Rho. Assuming that T_n is cyclic of prime order (i.e. the prime has size q^{n-1}), the Pollard–Rho algorithm performs $O(q^{(n-1)/2})$ steps, and each step consists essentially of a point addition and hence has complexity $\tilde{O}(1)$. Comparing this to the complexity of the index calculus algorithm in q, which is $\tilde{O}(q^{2-2/(n-1)})$, we see that the index calculus algorithm has smaller complexity for $n > 3$. More precisely, when $n = 3$ then Pollard–Rho and index calculus have the same complexity, when $n = 5$ the advantage of the index calculus attack comes only from the large prime variation (because without the large prime variation, index calculus has complexity $\tilde{O}(q^2)$), and when $n > 5$, the index calculus method is always better, independently of the large prime trick. The larger n, the larger the advantage of the index calculus algorithm over Pollard–Rho in this analysis.

However, the Pollard–Rho algorithm has to perform only an elliptic curve point addition in each step, while the index calculus algorithm has to compute a Gröbner basis, which is much more expensive. Even in the case $n = 3$, where the system is much more manageable than for larger n, we can solve less than a thousand systems per second (cf. Table 7.1), whereas elliptic curve point addition can be performed at a rate of 25000 to 150000 per second (depending on the size of the field; we measured this by adding random points of T_3 in Magma, an optimized implementation can achieve much better values). For larger values of n, the difference becomes much more extreme, since the cost of elliptic curve point addition increases only at the same rate as that of finite field arithmetic in $\mathbb{F}_{q^n}$, whereas the cost of the Gröbner basis computation increases considerably, since the degree of the equations grows exponentially and the number of equations and variables grows linearly in n. This becomes very obvious in our experiments, where we conclude that we cannot solve one single system for $n = 5$, and it is reflected also in the large complexity in n of the index calculus algorithm (see Theorem 7.8).

We conclude that in practice index calculus can beat Pollard–Rho only for moderate values of $n > 3$ and very large values of q. The precise crossover point is not known.

Notice also that the variant of the index calculus algorithm for T_5 that uses the trick of Joux and Vitse and the hybrid approach has complexity $\tilde{O}(q^3)$ in q, therefore it is not better than the Pollard–Rho algorithm for $n = 5$. It would be better only for $n > 5$.

Index calculus on the whole curve. The index calculus algorithm of Gaudry may also be used to compute discrete logarithms in $E(\mathbb{F}_{q^n})$ by working in the n-dimensional Weil restriction of E with respect to $\mathbb{F}_{q^n}|\mathbb{F}_q$. This is one of the original applications suggested by Gaudry in [Gau09]. From a complexity theoretic point of view, it does not make sense to attack the DLP in $E(\mathbb{F}_{q^n})$ when one wants to solve a DLP in T_n, since the complexity of Gaudry's algorithm in q depends on the dimension of the variety and therefore has complexity $\tilde{O}(q^{2-2/n})$ in $E(\mathbb{F}_{q^n})$ and complexity $\tilde{O}(q^{2-2/(n-2)})$ in T_n.

From a practical point of view, however, the systems one gets when performing index calculus on the whole curve may be more manageable, since they consist only of the Weil restriction of the Semaev polynomial, whereas in our approach, the system contains also the equations of the factor base. Moreover, when working in the whole curve, the Semaev polynomial may easily be symmetrized, which gives a system of smaller degree and with fewer solutions, whereas it is not obvious how to do this in our case. Also, when working in the whole curve, factor base elements may be represented by one $\mathbb{F}_q$-coordinate only, where we need two for the trace zero variety. Therefore, our system has twice as many indeterminates. On the other hand, the advantage of working in the trace zero variety is that relations contain $n-1$ factor base elements, and therefore one uses f_n to describe relations, whereas when working on the whole curve, relations contain n factor base elements, thus one has to use f_{n+1}. Summarizing, when working on the whole curve, one has a system of n equations in n indeterminates of total degree 2^{n-1}. In contrast, when working in the trace zero variety one has a system of $2n-1$ equations in $2n-2$ indeterminates of total degree $(n-1)2^{n-2}$.

Such subtleties are not evident in the original complexity analysis of Gaudry, which is only in q (and n is taken to be constant) and where the Gröbner basis computation thus has constant complexity. When performing an analysis similar to the one of Section 7.2 for Gaudry's algorithm on the whole curve, one obtains

$$\tilde{O}\left(\binom{n2^{n-1}+1}{n}^{\omega} n! q^{2-2/n}\right),$$

which is smaller in n. Therefore, which attack is better depends on the ratio of q and n.

It is difficult to determine the exact crossover point, since the cases where one can do experiments are rather limited. As an example, let us consider $n = 5$. In this case, the system one obtains for index calculus in the whole curve cannot be solved. For this reason, Joux and Vitse [JV12] propose searching for relations with $n-1$ factor base elements, and they can readily solve the system in this case, at the expense of multiplying the complexity by a factor q. When working in the trace zero variety, since it has dimension $n-1$ we have to search for relations involving $n-1$ factor base elements from the start. However, we cannot solve this system (compared to the one of Joux and Vitse, our system also comes from the n-th Semaev polynomial, but it has more equations, more indeterminates, and higher degree), which is why we consider relations involving $n-2$ factor base elements, also at the expense of a factor q in complexity. Unfortunately, we cannot solve this system either, which is why we use the hybrid method, which finally enables us to solve our system, at the expense of another factor q in the complexity.

All of this shows that it is difficult to accurately predict the running time of such index calculus algorithms and that the feasibility of the Gröbner basis computation plays an important role in practice.

Furthermore, the work of Diem [Die11], Faugère, Perret, Petit, and Renault [FPPR12], Petit and Quisquater [PQ12], and Shantz and Teske [ST13] suggests that the index calculus algorithm in $E(\mathbb{F}_{q^n})$ may even have sub-exponential complexity, since the polynomial systems that appear in this setting have a special shape, and special-purpose Gröbner basis techniques yield a significant speed-up. Some of these papers consider curves over characteristic 2 fields only.

Cover attacks. Cover attacks, also referred to as transfer attacks, were first proposed by Frey [Fre99] and further studied by many authors, including Galbraith and Smart [GS99], Gaudry, Hess, and Smart [GHS02], and Diem [Die03]. The aim of such attacks is to transfer the DLP from the algebraic variety one is considering into the Picard group of a curve of larger (but still rather low) genus, where the DLP is then solved using index calculus methods. There exist different constructions, each of them specific to a certain type of curve or variety, and there are constructions for cover attacks on $E(\mathbb{F}_{q^n})$ and on T_n directly.

For example, combining the results of [Die03] and [DK13], it is sometimes possible to map the DLP into the Picard group of a genus 5 curve (which is usually not hyperelliptic), where it can be solved in $\tilde{O}(q^{4/3})$. This is better than Gaudry's index calculus in $E(\mathbb{F}_{q^5})$, which has complexity $\tilde{O}(q^{8/5})$, and the index calculus attack on T_5, which has complexity $\tilde{O}(q^{3/2})$. However, the index calculus attack on T_5 applies to all curves, whereas only a very small proportion of curves is affected by the cover attack.

Diem and Scholten [DS, DS03] propose a cover attack for the trace zero variety directly. It works best for trace zero varieties of genus 2 curves, but it also applies to some trace zero varieties of elliptic curves. Namely, when $g = 1$ and $n = 5$, the DLP may sometimes be transferred to a curve of genus 4, where it can be solved in $\tilde{O}(q^{4/3})$. Again, this is better than the complexity of the index calculus attack, but it only affects a small number of curves (in fact, in [DS03] the authors find only one curve vulnerable to this attack). The same is true for $g = 1$ and $n = 7$, where the DLP may sometimes be mapped to a curve of genus 8 (and in this case, the authors cannot find any examples, although they can show that vulnerable curves exist in theory).

Joux and Vitse [JV12] propose a cover and decomposition attack, which combines a cover attack with index calculus. However, this only applies to elliptic curves over composite degree extension fields, and therefore it does not threaten the trace zero variety.

7.5 Conclusions on the hardness of the DLP

We conclude that applying Gaudry's index calculus algorithm for abelian varieties to the trace zero variety, as presented in this chapter, yields an attack in T_n that has smaller complexity than generic algorithms whenever $n > 3$ when the complexity is measured asymptotically in q. Although there sometimes exist cover attacks with even better complexity, the index calculus attack can be applied to trace zero varieties of all elliptic curves, while cover attacks apply only to a small proportion of curves.

Since the DLP in T_n has the same complexity as the DLP in $E(\mathbb{F}_{q^n})$, we get that the DLP in $E(\mathbb{F}_{q^n})$ may, in fact, be attacked in complexity $\tilde{O}(q^{2-2/(n-1)})$ when E is defined over $\mathbb{F}_q$. This is better than all known direct attacks on the DLP in $E(\mathbb{F}_{q^n})$ for $n > 2$. The most interesting case in this context is when $n = 3$. Here generic attacks on $E(\mathbb{F}_{q^3})$

have complexity $O(q^{3/2})$, Gaudry's index calculus attack applied to $E(\mathbb{F}_{q^3})$ has complexity $\tilde{O}(q^{4/3})$, and our index calculus attack on T_3 has complexity $\tilde{O}(q)$. Moreover, we have seen that our algorithm is practical for small q in this case, since the system has a particularly simple shape and can be solved rapidly. This becomes faster than Pollard–Rho for values of q larger than about 60 bits, and asymptotically, it yields one of the currently best attacks on the DLP in $E(\mathbb{F}_{q^3})$. Notice, however, that the Pollard–Rho algorithm in T_3 has the same complexity and is much faster in practice.

For general n, we have seen that the complexity of our index calculus attack on T_n depends exponentially on n and that it becomes infeasible for rather small values of n. This is due to the fact that the algorithm has to solve many polynomial systems, whose size (i.e. number of equations, number of indeterminates, degrees of the equations) depends on n, and that a Gröbner basis computation quickly becomes unmanageable. In fact, already for $n = 5$ it is impossible to solve the system with standard Gröbner basis software. By using some tricks (namely, considering relations that involve one point less, using a hybrid approach), we were nevertheless able to produce relations, but this does not yield a practical attack, since it multiplies the complexity of the relation search by a factor q^2.

Specialized Gröbner basis techniques in the spirit of [JV11, FPPR12, PQ12] would be needed in order to efficiently solve the systems that arise in this index calculus attack, and more research needs to be done on this topic in order to make our index calculus attack feasible in practice.

We finish with some remarks on the security of trace zero subgroups of elliptic curves for DLP-based cryptosystems for the practically most relevant cases of $n = 3, 5$.

Extension degree $n = 3$. To the extent of our knowledge (see also [ACD+06, Example 23.7]), there are no known attacks on the DLP in T_3 whose complexity is lower than that of generic (square root) attacks, provided that one chooses the curve according to usual cryptographic practice. In particular, the group should have prime or almost prime order and be sufficiently large (e.g. 160 or 200 bits). We stress that the index calculus attack, as detailed in this chapter, is not better than generic (square root) attacks in this setting, since the trace zero variety has dimension 2.

Extension degree $n = 5$. Since T_5 is a group of size q^4, generic attacks have complexity $O(q^2)$. Security threats are posed by algorithms for solving the DLP that achieve lower complexity. This is the case for the index calculus algorithm, which has complexity $\tilde{O}(q^{3/2})$. However, we have seen that it is not practical using standard Gröbner basis software. A more in depth study of specialized Gröbner basis methods is needed in order to determine to what extent the index calculus algorithm is a practical threat to T_5. As explained in Section 7.4, cover attacks can solve the DLP in $E(\mathbb{F}_{q^5})$ or T_5 in complexity $O(q^{4/3})$ for a small number of curves. Such curves should, of course, be avoided in trace zero cryptosystems.

Hyperelliptic curves. Although this chapter is not concerned with trace zero subgroups of Picard groups of hyperelliptic curves, we comment briefly on this case, since such groups have been proposed for cryptosystems. The parameters proposed by Lange [Lan04b, Lan01], namely genus 2 and $n = 3$, are threatened only by an index calculus attack in the same spirit as the one presented in this chapter, i.e. following Gaudry's approach. Since the trace zero variety has dimension 4 in this case, generic attacks have complexity $O(q^2)$, whereas the index calculus attack would have complexity $\tilde{O}(q^{3/2})$. However, the details of such an attack, and in particular the shape of the polynomial systems, has not yet been studied. For larger values of n and g, the gain of such an attack becomes more extreme, but the polynomial systems may very well be unmanageable. Moreover, for genus 2 and $n = 3$, a small proportion of trace zero subgroups is vulnerable to cover attacks, see [DS, DS03], and such curves must be avoided. Finally, it is well known that curves of larger genus may be attacked by sub-exponential index calculus algorithms (see e.g. [EGT11]), therefore using trace zero varieties associated to higher genus hyperelliptic curves in cryptosystems is not advisable.

Pairing-based cryptography. We finish by pointing out that—if the curve and the parameters are chosen carefully—the attacks on the DLP in T_n discussed above do not pose a threat to pairing-based cryptosystems based on the trace zero variety as suggested by Rubin and Silverberg in [RS09], where the authors show that supersingular abelian varieties of dimension greater than one offer more security than supersingular elliptic curves, for the same group size.

As an example, let us consider $n = 5$, which is mentioned as a particularly interesting case in [RS09]. In order to estimate the security of T_5 in pairing-based cryptosystems, one needs to compare the complexities of solving the DLP in T_5 and in $\mathbb{F}_{q^{5k}}$, where k is the embedding degree, i.e. the smallest integer k such that $\mathbb{F}_{q^{5k}}$ contains the image of the pairing. A first observation is that, since the results of [RS09] hold over fields of any characteristic, one should avoid fields of small characteristic, so that the recent attacks from [GGMZ13a, Jou13b, GGMZ13b, BBD$^+$14, BGJT13, AMOR13] do not apply. Over a field of large characteristic, the cover and index calculus attacks that we discussed in Section 7.4 do not seem to pose a serious security concern in the context of pairing-based cryptography. This is due to the fact that, for most supersingular elliptic curves, the Frey–Rück or the MOV attack [FR94, MOV93] have lower complexity than index calculus attacks and cover attacks. In some cases however, the choice of the security parameter may need to be adjusted, according to the complexity of these attacks.

As a concrete example, let us discuss the choice of parameters for a pairing with 80-bit security. One needs a field of about 1024 bits as the target of the pairing (avoiding fields of small characteristic). If we assume that the pairing maps to an extension field of degree $k = 2$ of the original field $\mathbb{F}_{q^5}$ (this is the case for most supersingular elliptic curves), then q should be a 102-bit number. A $q^{3/2}$ attack on the group T_5 on which the pairing is defined would result in 153-bit security, while a $q^{4/3}$ attack would result in 136-bit security. However, on the side of the finite field the system has 80-bit security, so the index calculus attack ends up not influencing the overall security of the pairing-based cryptosystem in this case.

A related comment is that an interesting case for pairings is when the DLP in T_5 and in the finite field extension $\mathbb{F}_{q^{5k}}$ where the pairing maps have the same complexity. In order to accomplish this in our previous example, we would need to have a security parameter $k = 4$, which can be achieved by supersingular trace zero varieties. In this case, a 53-bit q yields a complexity of about 80 bits for solving both a DLP in T_5 (with the $q^{3/2}$ attack) and a DLP in $\mathbb{F}_{q^{20}}$.

Summarizing, the complexity of index calculus and cover attacks on the DLP in T_5 influences the choice of the specific curves that we use in pairing-based applications, since it influences the security parameter k that makes the hardness of solving the DLP in T_5 and in $\mathbb{F}_{q^{5k}}$ comparable, and the value of k depends on the choice of the curve. However, in general it does not influence the size q of the field that we work with, since an attack can influence the value of q only if it has lower complexity than the Frey–Rück or the MOV attack for supersingular elliptic curves. Therefore, using trace zero varieties instead of elliptic curve groups in pairing-based cryptography has the advantages of enhancing the security and allowing for more flexibility in the setup of the system.

References

[AMOR13] G. Adj, A. Menezes, T. Oliveira, and F. Rodríguez-Henríquez. Weakness of $\mathbb{F}_{3^{6\cdot 509}}$ for discrete logarithm cryptography. In Z. Cao and F. Zhang, editors, *Pairing-Based Cryptography – Pairing 2013*, volume 8365 of *LNCS*, pages 20–44. Springer, 2013.

[Adl79] L. M. Adleman. A subexponential algorithm for discrete logarithms with applications to cryptography. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 55–60. IEEE, 1979.

[Adl94] L. M. Adleman. The function field sieve. In *Algorithmic Number Theory (ANTS I)*, volume 877 of *LNCS*, pages 108–121. Springer, 1994.

[ADH94] L. M. Adleman, J. DeMarrais, and M.-D. A. Huang. A subexponential algorithm for discrete logrithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields. In L. M. Adleman and M.-D. A. Huang, editors, *Algorithmic Number Theory (ANTS I)*, volume 877 of *LNCS*, pages 28–40. Springer, 1994.

[AFI$^+$04] G. Ars, J.-C. Faugère, H. Imai, M. Kawazoe, and M. Sugita. Comparison between XL and Gröbner basis algorithms. In P. J. Lee, editor, *Advances in Cryptology: Proceedings of ASIACRYPT '04*, volume 3329 of *LNCS*, pages 338–353. Springer, 2004.

[ACD$^+$06] R. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton, 2006.

[AC07] R. M. Avanzi and E. Cesena. Trace zero varieties over fields of characteristic 2 for cryptographic applications. In *Proceedings of the First Symposium on Algebraic Geometry and Its Applications (SAGA '07)*, pages 188–215, 2007.

[BFS04] M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of ICPSS '04*, pages 71–75, 2004.

[BFSY05] M. Bardet, J.-C. Faugère, B. Salvy, and B.-Y. Yang. Asymptotic behaviour of the index of regularity of quadratic semi-regular polynomial systems. In P. Gianni, editor, *The Effective Methods in Algebraic Geometry Conference (MEGA '05)*, pages 1–14, 2005.

[BV06] P. S. L. M. Barreto and J. S. Voloch. Efficient computation of roots in finite fields. *Des. Codes Crytogr.*, 39(2):275–280, 2006.

[BS88] D. Bayer and M. Stillman. On the complexity of computing syzygies. *J. Symbolic Comput.*, 6:135–147, 1988.

[BDL+12] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *J. Cryptogr. Eng.*, 2(2):77–89, 2012.

[BFP08] L. Bettale, J.-C. Faugère, and L. Perret. Hybrid approach for solving multivariate systems over finite fields. *J. Math. Cryptol.*, 2:1–22, 2008.

[Bla02] G. Blady. Die Weil-Restriktion elliptischer Kurven in der Kryptographie. Master's thesis, Univerität GHS Essen, 2002.

[Boe90] B. den Boer. Diffie–Hellman is as strong as discrete log for certain primes. In S. Goldwasser, editor, *Advances in Cryptology: Proceedings of CRYPTO '88*, volume 403 of *LNCS*, pages 530–539. Springer, 1990.

[BF03] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003.

[BCHL13] J. W. Bos, C. Costello, H. Hisil, and K. Lauter. High-performance scalar multiplication using 8-dimensional GLV/GLS decomposition. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *LNCS*, pages 331–338. Springer, 2013.

[BKK+09] J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery. Playstation 3 computing breaks 2^{60} barrier: 112-bit prime ECDLP solved. Available at http://lacal.epfl.ch/112bit_prime, 2009.

[BLR80] S. Bosch, W. Lütkebohmert, and M. Raynaud. *Néron Models*. Springer, Berlin–Heidelberg–New York, 1980.

[BCP97] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24:235–265, 1997.

[Bou12] C. Bouvier. The filtering step of discrete logarithm and integer factorization algorithms. Available at http://hal.inria.fr/hal-00734654, 2012.

[BBD+14] R. Bărbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, and P. Zimmermann. Discrete logarithm in GF(2^{809}) with FFS. In H. Krawczyk, editor, *Public-Key Cryptography – PKC 2014*, volume 8383 of *LNCS*, pages 221–238. Springer, 2014.

[BGJT13] R. Bărbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. To appear in Proceedings of EUROCRYPT '14, available at http://arxiv.org/abs/1306.4244, 2013.

[Buc65] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Leopold-Franzens-Univerisät Innsbruck, 1965.

[Can87] D. G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, 48(177):95–101, 1987.

[Ces08] E. Cesena. Pairing with supersingular trace zero varieties revisited. Available at http://eprint.iacr.org/2008/404, 2008.

[Ces10] E. Cesena. *Trace Zero Varieties in Pairing-based Cryptography.* PhD thesis, Università degli studi Roma Tre, Available at http://ricerca.mat.uniroma3.it/dottorato/Tesi/tesicesena.pdf, 2010.

[CoC] CoCoATeam. CoCoA: a system for doing computations in commutative algebra. Available at http://cocoa.dima.unige.it.

[CKM97] S. Collart, M. Kalkbrenner, and D. Mall. Converting bases with the Gröbner walk. *J. Symbolic Comput.*, 24:465–469, 1997.

[Cop84] D. Coppersmith. Fast evalution of logarithms in fields of characteristic two. *IEEE Trans. Inform. Theory*, 30:587–594, 1984.

[CLRS09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge–London, third edition, 2009.

[CKPS00] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In B. Preneel, editor, *Advances in Cryptology: Proceedings of EUROCRYPT '00*, volume 1807 of *LNCS*, pages 392–407. Springer, 2000.

[CLO92] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, Berlin–Heidelberg–New York, 1992.

[DGPS12] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann. Singular – A computer algebra system for polynomial computations. Available at http://www.singular.uni-kl.de, 2012.

[Die01] C. Diem. *A Study on Theoretical and Practical Aspects of Weil-Restrictions of Varieties*. PhD thesis, Univerität GHS Essen, Available at http://www.math.uni-leipzig.de/~diem/preprints, 2001.

[Die03] C. Diem. The GHS attack in odd characteristic. *Ramanujan Math. Soc.*, 18(1):1–32, 2003.

[Die06] C. Diem. An index calculus algorithm for plane curves of small degree. In F. Hess, S. Pauli, and M. Pohst, editors, *Algorithmic Number Theory (ANTS VII)*, volume 4076 of *LNCS*, pages 543–557. Springer, 2006.

[Die11] C. Diem. On the discrete logarithm problem in elliptic curves. *Compos. Math.*, 147:75–104, 2011.

[Die13] C. Diem. On the discrete logarithm problem in elliptic curves II. *Algebra & Number Theory*, 7:1281–1323, 2013.

[DK13] C. Diem and S. Kochinke. Computing discrete logarithms with special linear systems. Available at http://www.math.uni-leipzig.de/~diem/preprints, 2013.

[DS] C. Diem and J. Scholten. An attack on a trace-zero cryptosystem. Available at http://www.math.uni-leipzig.de/diem/preprints.

[DS03] C. Diem and J. Scholten. Cover attacks – A report for the AREHCC project. Available at http://www.math.uni-leipzig.de/~diem/preprints, 2003.

[DT08] C. Diem and E. Thomé. Index calculus in class groups of non-hyperelliptic curves of genus three. *J. Cryptology*, 21:593–611, 2008.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, 22(6):644–654, 1976.

[Duq04] S. Duquesne. Montgomery scalar multiplication for genus 2 curves. In D. Buell, editor, *Algorithmic Number Theory (ANTS VI)*, volume 3076 of *LNCS*, pages 153–168. Springer, 2004.

[DGM99] I. Duursma, P. Gaudry, and F. Morain. Speeding up the discrete log computation on curves with automorphisms. In K. Y. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology: Proceedings of ASIACRYPT '99*, volume 1716 of *LNCS*, pages 103–121. Springer, 1999.

[EGO11] P. N. J. Eagle, S. D. Galbraith, and J. Ong. Point compression for Koblitz curves. *Adv. Math. Commun.*, 5(1):1–10, 2011.

[EK97] E. Eberly and K. Kaltofen. On randomized Lanczos algorithms. In W. W. Küchlin, editor, *Proceedings of the 1997 international symposium on Symbolic and algebraic computation (ISSAC '97)*, pages 176–183. ACM, 1997.

[Eis04] D. Eisenbud. *Commutative algebra with a view towards algebraic geometry*, volume 150 of *Graduate Texts in Mathematics*. Springer, Berlin–Heidelberg–New York, 2004.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[Eng02] A. Enge. Computing discrete logarithms in high-genus hyperelliptic Jacobians in provably subexponential time. *Math. Comp.*, 71:729–742, 2002.

[Eng08] A. Enge. Computing discrete logarithms in curves over finite fields. In G. L. Mullen, D. Panario, and I. E. Shparlinski, editors, *Finite Fields Appl.*, volume 461 of *Contemp. Math.*, pages 119–139. AMS, 2008.

[EG02] A. Enge and P. Gaudry. A general framework for subexponential discrete logarithm algorithms. *Acta Arith.*, 102:83–103, 2002.

[EG07] A. Enge and P. Gaudry. An $L(1/3+\varepsilon)$ algorithm for the discrete logarithm problem for low degree curves. In M. Naor, editor, *Advances in Cryptology: Proceedings of EUROCRYPT '07*, volume 4515 of *LNCS*, pages 379–393. Springer, 2007.

[EGT11] A. Enge, P. Gaudry, and E. Thomé. An $L(1/3)$ discrete logarithm algorithm for low degree curves. *J. Cryptology*, 24:24–41, 2011.

[Fau99] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F4). *J. Pure Appl. Algebra*, 139(1):61–88, 1999.

[Fau02] J.-C. Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation (ISSAC '02)*, pages 75–83. ACM, 2002.

[FGHR12] J.-C. Faugère, P. Gaudry, L. Huot, and G. Renault. Using symmetries and fast change of ordering in the index calculus for elliptic curves discrete logarithm. In *Proceedings of the Third International Conference on Symbolic Computation and Cryptography (SCC '12)*, pages 113–118, 2012.

[FGHR13] J.-C. Faugère, P. Gaudry, L. Huot, and G. Renault. Using symmetries in the index calculus for elliptic curves discrete logarithm. To appear in J. Cryptology, Springer, DOI: 10.1007/s00145-013-9158-5, 2013.

[FGLM93] J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symbolic Comput.*, 16(4):329–344, 1993.

[FPPR12] J.-C. Faugère, L. Perret, C. Petit, and G. Renault. Improving the complexity of index calculus algorithms in elliptic curves over binary fields. In

D. Pointcheval and T. Johansson, editors, *Advances in Cryptology: Proceedings of EUROCRYPT '12*, volume 7237 of *LNCS*, pages 27–44. Springer, 2012.

[FHLS13] A. Faz-Hernández, P. Longa, and A. H. Sánchez. Efficient and secure algorithms for GLV-based scalar multiplication and their implementation on GLV-GLS curves. Available at http://eprint.iacr.org/2013/158, 2013.

[Fre99] G. Frey. Applications of arithmetical geometry to cryptographic constructions. In *Proceedings of the 5th International Conference on Finite Fields and Applications*, pages 128–161. Springer, 1999.

[FR94] G. Frey and H. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 62:865–874, 1994.

[Ful08] W. Fulton. *Algebraic Curves. An Introduction to Algebraic Geometry.* Available at http://www.math.lsa.umich.edu/~wfulton, 2008.

[GL09] S. D. Galbraith and X. Lin. Computing pairings using x-coordinates only. *Des. Codes Crytogr.*, 50(3):305–324, 2009.

[GLS11] S. D. Galbraith, X. Lin, and M. Scott. Endomorphisms for faster elliptic curve cryptography on a large class of curves. *J. Cryptology*, 24(3):446–469, 2011.

[GS99] S. D. Galbraith and N. P. Smart. A cryptographic application of Weil descent. In M. Walker, editor, *Cryptography and Coding. Proceedings of the 7th IMA International Conference*, volume 1746 of *LNCS*, pages 191–200. Springer, 1999.

[GS06] S. D. Galbraith and B. A. Smith. Discrete logarithms in generalized Jacobians. Available at http://uk.arxiv.org/abs/math.NT/0610073, 2006.

[GLV01] R. P. Gallant, R. J. Lambert, and S. A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In J. Kilian, editor, *Advances in Cryptology: Proceedings of CRYPTO '01*, volume 2139 of *LNCS*, pages 190–200. Springer, 2001.

[Gau00] P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. In B. Preneel, editor, *Advances in Cryptology: Proceedings of EUROCRYPT '00*, volume 1807 of *LNCS*, pages 19–34. Springer, 2000.

[Gau04] P. Gaudry. Discrete logarithm in elliptic curves over extension fields of small degree, Talk at the Workshop on Elliptic Curve Cryptography in Bochum, Germany. Available at http://cacr.uwaterloo.ca/conferences/2004/ecc2004/slides.html, 2004.

[Gau07] P. Gaudry. Fast genus 2 arithmetic based on Theta functions. *J. Math. Cryptol.*, 1:243–265, 2007.

[Gau09] P. Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *J. Symbolic Comput.*, 44(12):1690–1702, 2009.

[GHS02] P. Gaudry, F. Hess, and N.P. Smart. Constructive and destructive facets of Weil descent. *J. Cryptology*, 15(1):19–46, 2002.

[GS12] P. Gaudry and E. Schost. Genus 2 point counting over prime fields. *J. Symbolic Comput.*, 47(4):368–400, 2012.

[GTTD07] P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comp.*, 76:475–492, 2007.

[GvzG99] J. Gerhard and J. von zur Gathen. *Modern Computer Algebra.* Cambridge University Press, Cambridge, 1999.

[GGMZ13a] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel. On the function field sieve and the impact of higher splitting probabilities: Application to discrete logarithms in $\mathbb{F}_{2^{1971}}$. To appear in Proceedings of CRYPTO '13, available at http://eprint.iacr.org/2013/074, 2013.

[GGMZ13b] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel. Solving a 6120-bit DLP on a desktop computer. To appear in Proceedings of SAC '13, available at http://eprint.iacr.org/2013/306, 2013.

[GH99] G. Gong and L. Harn. Public-key cryptosystems based on cubic finite field extensions. *IEEE Trans. Inform. Theory*, 45(7):2601–2605, 1999.

[Gor93] D. M. Gordon. Discrete logarithms in GF(p) using the number field sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.

[Gor11] E. Gorla. Torus-based cryptography. In S. Jajodia and H. v. Tilborg, editors, *Encyclopedia of Cryptography*, pages 1306–1308. Springer, Berlin–Heidelberg–New York, 2nd edition, 2011.

[GJV10] R. Granger, A. Joux, and V. Vitse. New timings for oracle-assisted SDHP on the IPSEC Oakley 'well known group' 3 curve. NMBRTHRY list, available at https://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind1007&L=NMBRTHRY&P=R156&1=NMBRTHRY&9=A&J=on&d=No+Match%3BMatch%3BMatches&z=4, 2010.

[GV05] R. Granger and F. Vercauteren. On the discrete logarithm problem on algebraic tori. In V. Shoup, editor, *Advances in Cryptology: Proceedings of CRYPTO '05*, volume 3621 of *LNCS*, pages 66–85. Springer, 2005.

[GS] D. R. Grayson and M. E. Stillman. Macaulay2, a software system for research in algebraic geometry. Available at www.math.uiuc.edu/Macaulay2.

[HSS01] F. Hess, G. Seroussi, and N. P. Smart. Two topics in hyperelliptic cryptography. In S. Vaudenay and A. M. Youssef, editors, *Proceedings of SAC '01*, volume 2259 of *LNCS*, pages 181–189. Springer, 2001.

[Jel13] H. Jeljeli. Accelerating iterative SpMV for discrete logarithm problem using GPUs. Available at http://hal.inria.fr/hal-00734975, 2013.

[Jel14] H. Jeljeli. Resolution of linear algebra for the discrete logarithm problem using GPU and multi-core architectures. Available at http://hal.inria.fr/hal-00946895, 2014.

[Jou00] A. Joux. A one round protocol for tripartite Diffie–Hellman. In W. Bosma, editor, *Algorithmic Number Theory (ANTS IV)*, volume 1838 of *LNCS*, pages 385–393. Springer, 2000.

[Jou13a] A. Joux. Faster index calculus for the medium prime case: Application to 1175-bit and 1425-bit finite fields. In T. Johansson and P. Nguyen, editors, *Advances in Cryptology: Proceedings of EUROCRYPT '13*, volume 7881 of *LNCS*, pages 177–193. Springer, 2013.

[Jou13b] A. Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. To appear in Proceedings of SAC '13, available at http://eprint.iacr.org/2013/095, 2013.

[JL02] A. Joux and R. Lercier. The function field sieve is quite special. In C. Fieker and D. R. Kohel, editors, *Algorithmic Number Theory (ANTS V)*, volume 2369 of *LNCS*, pages 431–445. Springer, 2002.

[JL03] A. Joux and R. Lercier. Improvements to the general number field sieve for discrete logarithms in prime fields. A comparison with the Gaussian integer method. *Math. Comp.*, 72(242):953–976, 2003.

[JL06] A. Joux and R. Lercier. The function field sieve in the medium prime case. In S. Vaudenay, editor, *Advances in Cryptology: Proceedings of EUROCRYPT '06*, volume 4004 of *LNCS*, pages 254–270. Springer, 2006.

[JV11] A. Joux and V. Vitse. A variant of the F4 algorithm. In *Topics in cryptology CT-RSA 2011*, volume 6558 of *LNCS*, pages 356–375. Springer, 2011.

[JV12] A. Joux and V. Vitse. Elliptic curve discrete logarithm problem over small degree extension fields. Application to the static Diffie-Hellman problem on $E(\mathbb{F}_{q^5})$. To appear in J. Cryptology, Springer, DOI: 10.1007/s00145-011-9116-z, 2012.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48:203–209, 1987.

[Kob89] N. Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1(3):139–150, 1989.

[Kob91] N. Koblitz. CM-curves with good cryptographic properties. In J. Feigenbaum, editor, *Advances in Cryptology: Proceedings of CRYPTO '91*, volume 576 of *LNCS*, pages 179–287. Springer, 1991.

[KR00] M. Kreuzer and L. Robbiano. *Computational Commutative Algebra 1*. Springer, Berlin–Heidelberg–New York, 2000.

[KR05] M. Kreuzer and L. Robbiano. *Computational Commutative Algebra 2*. Springer, Berlin–Heidelberg–New York, 2005.

[LO90] B. A. LaMacchia and A. M. Odlyzko. Solving large sparse linear systems over finite fields. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology: Proceedings of CRYPTO '90*, volume 537 of *LNCS*, pages 109–133. Springer, 1990.

[LW54] S. Lang and A. Weil. Number of points of varieties in finite fields. *Amer. J. Math.*, 76(4):819–827, 1954.

[Lan01] T. Lange. *Efficient Arithmetic on Hyperelliptic Curves*. PhD thesis, Univerität GHS Essen, Available at http://www.hyperelliptic.org/tanja/preprints.html, 2001.

[Lan04a] T. Lange. Montgomery addition for genus two curves. In D. Buell, editor, *Algorithmic Number Theory (ANTS VI)*, volume 3076 of *LNCS*, pages 309–317. Springer, 2004.

[Lan04b] T. Lange. Trace zero subvarieties of genus 2 curves for cryptosystem. *Ramanujan Math. Soc.*, 19(1):15–33, 2004.

[Lan05] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Appl. Algebra Engrg. Comm. Comput.*, 15:295–328, 2005.

[LS04] T. Lange and M. Stevens. Efficient doubling for genus two curves over binary fields. In H. Handschuh and M. A. Hasan, editors, *Proceedings of SAC '04*, volume 3375 of *LNCS*, pages 170–181. Springer, 2004.

[Laz83] D. Lazard. Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations. In *Proceedings of EUROCAL '83*, volume 162 of *LNCS*, pages 146–156. Springer, 1983.

[LV00] A. K. Lenstra and E. R. Verheul. The XTR public key system. In M. Bellare, editor, *Advances in Cryptology: Proceedings of CRYPTO '00*, volume 1880 of *LNCS*, pages 1–19. Springer, 2000.

[LS12] P. Longa and F. Sica. Four-dimensional Gallant–Lambert–Vanstone scalar multiplication. In X. Wang and K. Sako, editors, *Advances in Cryptology: Proceedings of ASIACRYPT '12*, volume 7658 of *LNCS*, pages 718–739. Springer, 2012.

[Map] Maplesoft, a division of Waterloo Maple Inc. *Maple 16 ™*. Waterloo, Ontario.

[Mau94] U. Maurer. Towards the equivalence of breaking the Diffie–Hellman protocol and computing discrete logarithms. In Y. Desmedt, editor, *Advances in Cryptology: Proceedings of CRYPTO '94*, volume 839 of *LNCS*, pages 271–281. Springer, 1994.

[MM82] E. W. Mayr and A. R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Adv. Math.*, 46:305–329, 1982.

[MOV93] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Inform. Theory*, 39:1639–1646, 1993.

[MWZ98] A. Menezes, Y. Wu, and R. Zuccherato. An elementary introduction to hyperelliptic curves. Appendix in Algebraic Aspects of Cryptography by N. Koblitz, pages 155–178. Springer, Berlin–Heidelberg–New York, 1998.

[Mil85] V. S. Miller. Use of elliptic curves in cryptography. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology: Proceedings of CRYPTO '85*, volume 218 of *LNCS*, pages 417–426. Springer, 1985.

[Mil04] V. S. Miller. The Weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.

[MW10] A. Montes and M. Wimber. Gröbner bases for polynomial systems with parameters. *J. Symbolic Comput.*, 45(12):1391–1425, 2010.

[Mon87] P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Math. Comp.*, 48(177):243–264, 1987.

[Nag04] K. Nagao. Improvement of Thériault algorithm of index calculus of Jacobian of hyperelliptic curves of small genus. Available at `http://eprint.iacr.org/2004/161`, 2004.

[Nag10] K. Nagao. Decomposition attack for the Jacobian of a hyperelliptic curve over an extension field. In G. Hanrot, F. Morain, and E. Thomé, editors, *Algorithmic Number Theory (ANTS IX)*, volume 6197 of *LNCS*, pages 285–300. Springer, 2010.

[Nat94] National Institute of Standards and Technology. Digital signature standard. Federal Information Processing Standard (FIPS) Publication 186, 1994.

[Nau99] N. Naumann. Weil-Restriktion abelscher Varietäten. Master's thesis, Univerität GHS Essen, Available at http://web.iem.uni-due.de/ag/numbertheory/dissertationen, 1999.

[Nec94] V. I. Nechaev. On the complexity of a deterministic algorithm for a discrete logarithm. *Math. Zametki*, 55:91–101, 1994.

[OLAR13] T. Oliveira, J. López, D. F. Aranha, and F. Rodríguez-Henríquez. Lambda coordinates for binary elliptic curves. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems – CHES 2013*, volume 8086 of *LNCS*, pages 311–330. Springer, 2013.

[PQ12] C. Petit and J. Quisquater. On polynomial systems arising from a Weil descent. In X. Wang and K. Sako, editors, *Advances in Cryptology: Proceedings of ASIACRYPT '12*, volume 7658 of *LNCS*, pages 451–466. Springer, 2012.

[PH78] S. C. Pohlig and M. E. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Trans. Inform. Theory*, 24:106–110, 1978.

[Pol78] J. M. Pollard. Monte Carlo methods for index computation (mod p). *Math. Comp.*, 32:918–924, 1978.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Commun. ACM*, 21:120–126, 1978.

[RS02] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In M. Yung, editor, *Advances in Cryptology: Proceedings of CRYPTO '02*, volume 2442 of *LNCS*, pages 336–353. Springer, 2002.

[RS03] K. Rubin and A. Silverberg. Torus-based cryptography. In D. Boneh, editor, *Advances in Cryptology: Proceedings of CRYPTO '03*, volume 2729 of *LNCS*, pages 349–365. Springer, 2003.

[RS09] K. Rubin and A. Silverberg. Using abelian varieties to improve pairing-based cryptography. *J. Cryptology*, 22(3):330–364, 2009.

[SOK01] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing over elliptic curve (in Japanese). Presented at The 2001 Symposium on Cryptography and Information Security, Oiso, Japan, 2001.

[Sch02] O. Schirokauer. The special function field sieve. *SIAM J. Discrete Math.*, 16:81–98, 2002.

[Sch89] C.-P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology: Proceedings of CRYPTO '89*, volume 435 of *LNCS*, pages 239–252, Berlin–Heidelberg–New York, 1989. Springer.

[Sch85] R. Schoof. Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, 44(170):483–494, 1985.

[Sem04] I. Semaev. Summation polynomials of the discrete logarithm problem on elliptic curves. Available at http://eprint.iacr.org/2004/031, 2004.

[ST13] M. Shantz and E. Teske. Solving the elliptic curve discrete logarithm problem using Semaev polynomials, Weil descent and Gröbner basis methods – an experimental study. Available at http://eprint.iacr.org/2013/596, 2013.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In W. Fumy, editor, *Advances in Cryptology: Proceedings of EUROCRYPT '97*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997.

[Sil05] A. Silverberg. Compression for trace zero subgroups of elliptic curves. *Trends Math.*, 8:93–100, 2005.

[Sil09] J. H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, Berlin–Heidelberg–New York, second edition, 2009.

[SS95] P. Smith and C. Skinner. A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. In J. Pieprzyk and R. Safavi-Naini, editors, *Advances in Cryptology: Proceedings of ASIACRYPT '94*, volume 917 of *LNCS*, pages 357–364. Springer, 1995.

[Sta04] C. Stahlke. Point compression on Jacobians of hyperelliptic curves over $\mathbb{F}_q$. Available at http://eprint.iacr.org/2004/030, 2004.

[Sti93] H. Stichtenoth. *Algebraic Function Fields and Codes*. Springer, Berlin–Heidelberg–New York, 1993.

[Sti06] D. R. Stinson. *Cryptography. Theory and Practice*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton, third edition, 2006.

[Thé03] N. Thériault. Index calculus attack for hyperelliptic curves of small genus. In C. S. Laih, editor, *Advances in Cryptology: Proceedings of ASIACRYPT '03*, volume 2894 of *LNCS*, pages 75–92. Springer, 2003.

[Tra88] C. Traverso. Gröbner trace algorithms. In *Proceedings of the 1988 international symposium on Symbolic and algebraic computation (ISSAC '88)*, volume 358 of *LNCS*, pages 125–138. Springer, 1988.

[VJS14] M. D. Velichka, M. J. Jacobson, Jr., and A. Stein. Computing discrete logarithms in the Jacobian of high-genus hyperelliptic curves over even characteristic finite fields. *Math. Comp.*, 83(286):935–963, 2014.

[Was08] L. C. Washington. *Elliptic Curves: Number Theory and Cryptography*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton–London–New York, second edition, 2008.

[Wat69] W. C. Waterhouse. Abelian varieties over finite fields. *Ann. Sci. Éc. Norm. Supér. (4)*, 2(4):521–560, 1969.

[Wei58] A. Weil. The field of definition of a variety. *Amer. J. Math.*, 78:509–524, 1958.

[Wei01] A. Weimerskirch. The application of the Mordell–Weil group to cryptographic systems. Master's thesis, Worcester Polytechnic Institute, Available at http://www.emsec.rub.de/media/crypto/attachments/files/2010/04/ms_weika.pdf, 2001.

[Wei92] V. Weispfenning. Comprehensive Gröbner bases. *J. Symbolic Comput.*, 14:1–29, 1992.

[Wie86] D. H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inform. Theory*, IT-32(1):54–62, 1986.

[YC04a] B.-Y. Yang and J.-M. Chen. All in the XL family: Theory and practice. In C. Park and S. Chee, editors, *Information Security and Cryptology (ICISC '04)*, volume 3506 of *LNCS*, pages 67–86. Springer, 2004.

[YC04b] B.-Y. Yang and J.-M. Chen. Theoretical analysis of XL over small fields. In H. Wang, J. Pieprzyk, and V. Varadharajan, editors, *Information Security and Privacy (ACISP '04)*, volume 3108 of *LNCS*, pages 277–288. Springer, 2004.

[YCC04] B.-Y. Yang, J.-M. Chen, and N. T. Courtois. On asymptotic security estimates in XL and Gröbner bases-related algebraic cryptanalysis. In J. López, S. Qing, and E. Okamoto, editors, *Information and Communications Security (ICICS '04)*, volume 3269 of *LNCS*, pages 401–413. Springer, 2004.

[YCY13] J. Y.-C. Yeh, C.-M. Cheng, and B.-Y. Yang. Operating degrees for XL vs. F_4/F_5 for generic $\mathcal{MQ}$ with number of equations linear in that of variables. In M. Fischlin and S. Katzenbeisser, editors, *Number Theory and Cryptography (Buchmann Festschrift)*, volume 8260 of *LNCS*, pages 19–33. Springer, 2013.

Articles

The following articles include material from this thesis.

E. Gorla and M. Massierer. Point compression for the trace zero subgroup over a small degree extension field. To appear in Des. Codes Cryptogr., Springer, DOI: 10.1007/s10623-014-9921-0, 2014.

E. Gorla and M. Massierer. An optimal representation for the trace zero variety. Preprint, 2013.

E. Gorla and M. Massierer. Index calculus in the trace zero variety. Preprint, 2014.

Curriculum vitae

Personal information. Maike Kerstin Massierer, born in Nürnberg (Germany)

Education.

12/2013 **Ph.D.** in mathematics
Universität Basel (Switzerland)
Thesis: *Trace zero varieties in cryptography: optimal representation and index calculus*
Advisor: Prof. Dr. Elisa Gorla

08/2009 **Diplom** (equivalent to **M.Sc.**) in mathematics (major) and computer science (minor)
Technische Universität Berlin (Germany)
Thesis: *Class field theory for global function fields and applications*
Advisor: Prof. Dr. Florian Heß

12/2006 Study abroad program equivalent to the degree
B.Sc. (Hons) in pure mathematics and computer science
University of New South Wales, Sydney (Australia)
Thesis: *Provably secure cryptographic hash functions*
Advisors: Dr. Richard Buckland and Prof. Dr. James Franklin

04/2005 **Vordiplom** in mathematics (major) and computer science (minor)
Friedrich–Alexander–Universität Erlangen–Nürnberg (Germany)

07/2003 **Abitur**
Goethe Gymnasium Ludwigsburg (Germany)

Talks. SIAM conference on applied algebraic geometry, Colorado State University (USA), Workshop algebra and geometry, Universität Bern (Switzerland), Basel–Dublin–Zürich workshop on crypto and coding, Universität Zürich (Switzerland), CrossFyre, Katholieke Universiteit Leuven (Belgium), 8th Swiss graduate colloquium, Universität Basel (Switzerland); seminars at Université de Bordeaux (France), École polytechnique (France), INRIA Nancy (France), Université de Versailles (France), Universität Oldenburg (Germany), Universität Zürich (Switzerland), Universität Basel (Switzerland), Technische Univerität Berlin (Germany)

Workshops, conferences, schools. ECC 2013, Katholieke Universiteit Leuven (Belgium), Tutorials in the mountains on numerical algebraic geometry, Colorado State University (USA), Oberwolfach seminar on algorithms for complex multiplication over finite fields, Mathematisches Forschungsinstitut Oberwolfach (Germany), Trends in coding theory, Ascona (Switzerland), Game theory, evolutionary game theory, and learning in games, Les Diablerets (Switzerland), ECRYPT II summer school on tools, Mykonos (Greece), SP-ASCrypto, Universidade Estadual de Campinas (Brazil), ECC 2011, INRIA Nancy (France), ACAGM summer school, Katholieke Universiteit Leuven (Belgium), ECC 2010, Microsoft Research, Redmond (USA), b-it summer school on cryptography, Bonn–Aachen International Center for Information Technology (Germany), 10. Kryptotag, Technische Universität Berlin (Germany), ECRYPT II winter school on mathematical foundations in cryptography, École Polytechnique Fédérale de Lausanne (Switzerland), ECM-EM RNSA workshop on recent advances in stream ciphers and hash functions, Queensland University of Technology (Australia)