Point compression for the trace zero subgroup over a small degree extension field*

Elisa Gorla[1] and Maike Massierer[2]

[1]Institut de mathématiques, Université de Neuchâtel, Rue Emile-Argand 11, 2000 Neuchâtel, Switzerland, elisa.gorla@unine.ch

[2]Mathematisches Institut, Universität Basel, Rheinsprung 21, 4051 Basel, Switzerland, maike.massierer@unibas.ch

Abstract Using Semaev's summation polynomials, we derive a new equation for the $\mathbb{F}_q$-rational points of the trace zero variety of an elliptic curve defined over $\mathbb{F}_q$. Using this equation, we produce an optimal-size representation for such points. Our representation is compatible with scalar multiplication. We give a point compression algorithm to compute the representation and a decompression algorithm to recover the original point (up to some small ambiguity). The algorithms are efficient for trace zero varieties coming from small degree extension fields. We give explicit equations and discuss in detail the practically relevant cases of cubic and quintic field extensions.

Keywords elliptic curve cryptography, pairing-based cryptography, discrete logarithm problem, trace zero variety, efficient representation, point compression, summation polynomials

Mathematics Subject Classification 14G50, 11G25, 14H52, 11T71, 14K15

1 Introduction

Given a (hyper)elliptic curve defined over $\mathbb{F}_q$ and a field extension $\mathbb{F}_q|\mathbb{F}_{q^n}$, consider the $\mathbb{F}_{q^n}$-rational points of trace zero. They form a subgroup of the group of $\mathbb{F}_{q^n}$-rational points of the curve, and can be realized as the $\mathbb{F}_q$-rational points of an abelian variety built by Weil restriction from the original curve, called the trace zero variety. The trace zero subgroup was first proposed for use in cryptography by Frey [15], and further studied by Naumann [35], Weimerskirch [44], Blady [5], Lange [31, 32], Avanzi–Cesena [1, 8], and Diem-Scholten [12]. Trace zero subgroups are interesting because they allow efficient arithmetic, due to a speed-up of the standard scalar multiplication using the Frobenius endomorphism. This is analogous to the use of endomorphisms to speed up scalar multiplication on Koblitz curves (see [30]) and GLV–GLS curves (see [19, 17]), which are the basis for several recent implementation speed records for elliptic curve arithmetic (see [34, 14, 6]).

The trace zero subgroup is of interest in the context of pairing-based cryptography. Rubin and Silverberg have shown in [37, 40] that the security of pairing-based cryptosystems can be improved by using 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.

*This article appeared in *Designs, Codes and Cryptography*, the final publication is available at http://link.springer.com/article/10.1007%2Fs10623-014-9921-0.

Since the trace zero subgroup is contained in the group of $\mathbb{F}_{q^n}$-rational points of the (Jacobian of the) curve, the DLP in the trace zero subgroup is at most as hard as the DLP in the curve. It is easy to show that in fact the DLP's in the two groups have the same complexity. From a mathematical point of view therefore, trace zero variety cryptosystems may be regarded as the (hyper)elliptic curve analog of torus-based cryptosystems such as LUC [43], Gong–Harn [25], XTR [33], and CEILIDIH [38].

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. Such an optimal-size representation consists of $\log_2 N$ bits, where N is the size of the group. See also [26] for a discussion on the significance of compact representations.

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. Notice moreover that storing the sign of the y-coordinate is unnecessary, since this representation is compatible with scalar multiplication of points: For any $k \in \mathbb{Z}$, the x-coordinates of the points kP and $-kP$ coincide.

The trace zero variety of an elliptic curve with respect to a *prime* extension degree n has dimension $n-1$, and we are interested in the $\mathbb{F}_q$-rational points. Hence, an optimal representation should have $\log_2 q^{n-1}$ bits, or consist of $n-1$ 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"). Naumann [35], Rubin–Silverberg [37, 39, 42], and Lange [32] propose compact representations with compression and decompression algorithms for genus 1 and genus 2 curves, respectively. The work by Eagle–Galbraith–Ong [13] on point compression methods for Koblitz curves is also related.

In this paper, we concentrate on extension fields of degree $n = 3$ or 5. This is due to the fact that an index calculus attack [20] and a cover attack [9, 10, 11] apply to T_n, making it vulnerable for large values of n. In this work we briefly discuss these attacks and come to the conclusion that there are no security issues for $n = 3$. For $n = 5$ the cover attacks can be avoided by imposing extra conditions, and the known index calculus attacks do not threaten the security of pairing-based cryptosystems involving trace zero subgroups of supersingular curves.

The main purpose of this paper is introducing a new representation for the points on the trace zero variety of an elliptic curve. The compression and decompression algorithms are more efficient than that of [42], 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 Diffie–Hellman-based cryptographic protocols.

The paper is structured as follows: In Section 2, we fix the notation, give the relevant definitions, and briefly recall the standard representation for points on the trace zero variety. We also discuss the simple case of the trace zero variety for a quadratic field extension. Using Semaev's summation polynomials, in Section 3 we derive 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 1 and

Proposition 4). In Section 4, using the equation that we produced in the previous section, 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 Sections 5 and 6 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 and comment on security issues for these parameters. It is generally agreed that 3 and 5 are the practically relevant extension degrees in the case of elliptic curves (see e.g. [32]).

2 Preliminaries

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 consider the group $E(\mathbb{F}_{q^n})$ of $\mathbb{F}_{q^n}$-rational points of E for field extensions of *prime* degree n. The group operation is point addition, and the neutral element is the point at infinity, denoted by $\mathcal{O}$. We denote indeterminates by lower case letters and finite field elements by upper case letters.

Definition 1. The Frobenius endomorphism on E is defined by

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

One can define a trace map

$$\mathrm{Tr} : E(\mathbb{F}_{q^n}) \mapsto E(\mathbb{F}_q),\ P \mapsto P + \varphi(P) + \varphi^2(P) + \ldots + \varphi^{n-1}(P),$$

relative to the field extension $\mathbb{F}_{q^n}|\mathbb{F}_q$. The kernel of the trace map is the *trace zero subgroup* of $E(\mathbb{F}_{q^n})$, which we denote by T_n.

By the process of Weil restriction, the points of T_n can be viewed as the $\mathbb{F}_q$-rational points of an abelian variety V of dimension $n-1$ defined over $\mathbb{F}_q$. V is called the *trace zero variety*.

In trace zero subgroups, arithmetic can be made more efficient by using the Frobenius endomorphism, following a similar approach to Koblitz curves and GLV–GLS curves. They turn out to be extremely interesting in the context of pairing-based cryptography, where they achieve the largest security parameters in some cases, as discussed in [37, 40, 1, 8].

It is easy to show that the DLP in $E(\mathbb{F}_{q^n})$ is as hard as the DLP in T_n. An explanation is given in [27] for the analogous case of algebraic tori: The trace maps a DLP in $E(\mathbb{F}_{q^n})$ to a DLP in $E(\mathbb{F}_q)$. By solving it in the smaller group, the discrete logarithm is obtained modulo the order of $E(\mathbb{F}_q)$. The remaining modular information required to compute the full discrete logarithm comes from solving a DLP in T_n. A formal argument, which applies to any short exact sequence of algebraic groups, is given in [18].

Proposition 1. *Consider the exact sequence*

$$0 \longrightarrow T_n \longrightarrow E(\mathbb{F}_{q^n}) \xrightarrow{\mathrm{Tr}} E(\mathbb{F}_q) \longrightarrow 0.$$

Then solving a DLP in $E(\mathbb{F}_{q^n})$ has the same complexity as solving a DLP in T_n and a DLP in $E(\mathbb{F}_q)$.

In the conclusions of [1], large bandwidth is mentioned as the only drawback of using trace zero subgroups in pairing-based cryptography. In this paper we solve this problem by finding an optimal representation for the elements of the trace zero subgroup.

Definition 2. Let G be a finite set. A *representation* for the elements of G is a bijection between G and a set of binary strings of fixed length ℓ. Equivalently, it is an injective map from G to $\mathbb{F}_2^\ell$. A representation is *optimal* if $|G| \sim 2^\ell$, i.e. if we need approximately $\log_2 |G|$ bits to represent an element of G.

Abusing terminology, in this paper we call representation a map from G to $\mathbb{F}_2^\ell$ 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 the representation *identifies* classes of at most d elements, namely those that have the same representation. Notice that the number of classes is about $|G|/d \sim |G|$, if d is a small constant.

Remark 1. Since the elements of a finite field $\mathbb{F}_q$ can be represented via binary strings of length $\log_2 q$, a representation for G can be given via a bijection between G and a subset of $\mathbb{F}_q^m$, for some m and some prime power q. Such a representation is optimal if and only if $|G| \sim q^m$.

Representing points of an elliptic curve via their x-coordinate is a standard example of optimal representation.

Example 1. It is customary to represent a point $(X, Y) \in E(\mathbb{F}_{q^n})$ via its x-coordinate $X \in \mathbb{F}_{q^n}$. The y-coordinate can then be recovered, up to sign, from the curve equation. If desired, the sign can be stored in one extra bit of information. Such a representation is optimal, since by Hasse's Theorem $|E(\mathbb{F}_{q^n})| \sim q^n$.

The representation from the previous example identifies pairs of points, since P and $-P$ have the same x-coordinate. We often say that the x-coordinate is a representation for the *equivalence class* consisting of P and $-P$. The representation that we propose in this paper identifies a small number of points as well. Before we discuss our representation, we notice that representing a point $P \in T_n$ via its x-coordinate is no longer optimal.

Remark 2. Since a point $P = (X, Y) \in T_n$ is an element of $E(\mathbb{F}_{q^n})$, we can represent P via $X \in \mathbb{F}_{q^n}$. Choosing an $\mathbb{F}_q$-basis of $\mathbb{F}_{q^n}$, we can represent $X \in \mathbb{F}_{q^n}$ as an n-tuple $(X_0, \ldots, X_{n-1}) \in \mathbb{F}_q^n$. Representing $P \in T_n$ as $X \in \mathbb{F}_{q^n}$ or as $(X_0, \ldots, X_{n-1}) \in \mathbb{F}_q^n$ however is not optimal, since $|T_n| \sim q^{n-1}$.

In this paper we find a representation for the elements of T_n, via $n-1$ coordinates in $\mathbb{F}_q$. Our representation is not injective, but it identifies a small number of points. Our approach is the following: We start from the representation of $P \in T_n$ as an n-tuple $\rho(P) = (X_0, \ldots, X_{n-1}) \in \mathbb{F}_q^n$, and write an equation in $\mathbb{F}_q[x_0, \ldots, x_{n-1}]$ which vanishes on $\rho(P)$ for all $P \in T_n$. This allows us to drop one coordinate of $\rho(P)$ and reconstruct it using the equation. Therefore, we can represent elements of T_n via $n-1$ coordinates in $\mathbb{F}_q$, which is optimal.

We now fix some notation that we will use when writing explicit equations in Sections 4, 5, and 6. Let $\mathbb{F}_q$ be the finite field with q elements, n a prime. For the sake of concreteness, we assume that $n \mid q-1$. Due to its simplicity, we always consider this case when writing explicit equations. All of our arguments however work for any n and q, see also Remark 3. 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 basis of the field extension.

When doing Weil restriction, we associate n new variables $x_0, \ldots, x_{n-1}$ to the variable x. They are related via

$$x = x_0 + x_1\zeta + \ldots + x_{n-1}\zeta^{n-1}. \tag{1}$$

We abuse terminology and use the term *Weil restriction* not only for the variety, but also for the process of writing equations for the Weil restriction. In particular for us, Weil restriction is a procedure that can be applied to a polynomial defined over $\mathbb{F}_{q^n}$ and results in n polynomials with n times as many variables, and coefficients in $\mathbb{F}_q$.

Remark 3. 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

$$x = x_0\alpha + x_1\alpha^q + \ldots + x_{n-1}\alpha^{q^{n-1}}.$$

It is easy to show that the case $n = 2$ allows a trivial optimal representation for the elements of T_n. Hence in the next sections we concentrate on the more interesting case of *odd* primes n.

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

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

In particular, representing a point $(X, Y) \in T_2$ by $X \in \mathbb{F}_q$ yields a representation of optimal size.

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 $2P = \mathcal{O}$, hence $P \in E[2](\mathbb{F}_q)$. If $P \notin E(\mathbb{F}_q)$, then $(X, Y) = -(X^q, Y^q)$. 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 $2P = \mathcal{O}$, 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 $(X, Y) + \varphi(X, Y) = \mathcal{O}$ and $(X, Y) \in T_2$. □

The next proposition will be useful when writing equations for the $\mathbb{F}_q$-rational points of the trace zero variety. For a multivariate polynomial h, we denote by $\deg_{x_i}(h)$ the degree of h in the variable x_i.

Proposition 3. *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 $x_0^q - x_0, \ldots, x_{n-1}^q - x_{n-1}$ with respect to any term order are $x_0^q, \ldots, x_{n-1}^q$, in particular they are pairwise coprime. Hence the polynomials $x_0^q - x_0, \ldots, x_{n-1}^q - x_{n-1}$ 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 $x_0^q - x_0, \ldots, x_{n-1}^q - x_{n-1}$, hence $h = 0$. □

3 An equation for the trace zero subgroup

In this section we use Semaev's summation polynomials [41] to write an equation for the set of $\mathbb{F}_q$-rational points of the trace zero variety. The equation involves the x-coordinates only and will help us in finding a better representation for the elements of the trace zero subgroup.

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

Definition 3. 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

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

with coefficients $A, B \in \mathbb{F}_q$.

Define the m-th summation polynomial f_m recursively by

$$\begin{aligned} f_3(z_1,z_2,z_3) &= (z_1-z_2)^2 z_3^2 - 2((z_1+z_2)(z_1z_2+A)+2B)z_3 + (z_1z_2-A)^2 - 4B(z_1+z_2) \\ f_m(z_1,\ldots,z_m) &= \mathrm{Res}_z(f_{m-k}(z_1,\ldots,z_{m-k-1},z), f_{k+2}(z_{m-k},\ldots,z_m,z)) \end{aligned}$$

for $m \geq 4$ and $m-3 \geq k \geq 1$.

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

Theorem 1 ([41], Theorem 1)**.** *For any $m \geq 3$, 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 $(Z_1,Y_1)+\ldots+(Z_m,Y_m) = \mathcal{O}$ in the group $E(\overline{\mathbb{F}}_q)$. Furthermore, f_m is absolutely irreducible and symmetric of degree 2^{m-2} in each variable. The total degree is $(m-1)2^{m-2}$.*

Remark 4. Definition 3 is the original definition that Semaev gave in [41]. 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 1. Hence all the results that we prove in this paper hold, with the appropriate adjustments, over a finite field of any charasteristic.

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 1. *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*

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

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

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

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

Notice that the points of the form $P = Q + R$ 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 4. *Let T_n be the trace zero subgroup associated with the field extension $\mathbb{F}_{q^n}|\mathbb{F}_q$. We have*

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

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 $(X, Y_0) + (X^q, Y_1) + (X^{q^2}, Y_2) = \mathcal{O}$. Obviously we have $Y_i = \pm Y^{q^i}, i = 0, 1, 2$, so $P \pm \varphi(P) \pm \varphi^2(P) = \mathcal{O}$. We have to show that all signs are "+". Suppose $P - \varphi(P) + \varphi^2(P) = \mathcal{O}$. By applying φ, we get $\varphi(P) - \varphi^2(P) + P = \mathcal{O}$. Adding these two equations gives $2P = \mathcal{O}$, implying that $P = -P$, hence $P + \varphi(P) + \varphi^2(P) = \mathcal{O}$. 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 "+", then $P \in T_5$. We treat all other cases below.

[one minus] Assume $P + \varphi(P) + \varphi^2(P) + \varphi^3(P) - \varphi^4(P) = \mathcal{O}$. Applying φ to the equation and adding the two equations, we get $2\varphi(P) + 2\varphi^2(P) + 2\varphi^3(P) = \mathcal{O}$, and by substituting into twice

the first equation, $2P = \varphi^4(2P)$. Hence $2P \in E(\mathbb{F}_{q^4}) \cap E(\mathbb{F}_{q^5}) = E(\mathbb{F}_q)$, so $2P \in E[3](\mathbb{F}_q)$. Now $P = Q + R \in E[6]$ is the sum of $Q \in E[3]$ and $R \in E[2]$. We have $Q = -2Q = -2P \in E[3](\mathbb{F}_q)$. From the original equation $P+\varphi(P)+\varphi^2(P)+\varphi^3(P)-\varphi^4(P) = \mathcal{O}$, 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 $P + \varphi(P) + \varphi^2(P) - \varphi^3(P) - \varphi^4(P) = \mathcal{O}$. Applying φ^2 and adding, we get $2\varphi^2(P) = \mathcal{O}$, hence $P = -P$ and therefore $P \in T_5$.

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

The other cases follow by symmetry, thus proving the claim. □

Remark 5. In the sequel, we use f_n as an equation for T_n. In practice however, for any root $X \in \mathbb{F}_{q^n}$ of $f_n(x, x^q, \ldots, x^{q^{n-1}})$ we need to be able to 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 4.

For $n = 5$, by Proposition 4 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 $Q + R$ 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 $Q+R \in E[3](\mathbb{F}_q)+E[2] \cap T_5$ 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 $Y \in \mathbb{F}_{q^5}$.

The x-coordinates of the points of T_n correspond to zeros of the Weil restriction of the polynomial $f_n(x, \ldots, x^{q^{n-1}})$. Since E is defined over $\mathbb{F}_q$, then $f_n(x, \ldots, x^{q^{n-1}}) \in \mathbb{F}_q[x]$. Therefore, for any $\alpha \in \mathbb{F}_{q^n}$ we have

$$f_n(\alpha, \ldots, \alpha^{q^{n-1}})^q = f_n(\alpha^q, \ldots, \alpha^{q^{n-1}}, \alpha) = f_n(\alpha, \ldots, \alpha^{q^{n-1}}),$$

where the second equality follows from the symmetry of the Semaev polynomial. It follows that

$$f_n(\alpha, \ldots, \alpha^{q^{n-1}}) \in \mathbb{F}_q \quad \text{for all } \alpha \in \mathbb{F}_{q^n}. \tag{2}$$

We use the relations (1) to write equations for the Weil restriction. Notice that since we are only interested in the $\mathbb{F}_q$-rational points of the Weil restriction, we may reduce the equations that we obtain modulo $x_i^q - x_i$ for $i = 0, \ldots, n-1$. Hence we obtain equations in $x_0, \ldots, x_{n-1}$ of degree less than q in each indeterminate. Now (2) together with Proposition 3 implies that the last $n-1$ equations are identically zero. 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

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

We stress that its $\mathbb{F}_q$-solutions correspond to the elements of T_n, together with some extra points described in Lemma 1 and Proposition 4. In Remark 5 we discussed how to distinguish the extra solutions. Since we reduce the Weil restriction of $f_n(x, x^q, \ldots, x^{q^{n-1}})$ modulo $x_i^q - x_i$, the qth powers disappear, and we are left with an equation $\tilde{f}_n$ of the same degree as the original Semaev polynomial f_n.

Concerning the representation, we now have an equation that is compatible with dropping the y-coordinate. It is a natural idea to drop one X_i in order to obtain a compact representation, mimicking the approach of [35, 32, 42]. The decompression algorithm could then use $\tilde{f}_n$ to recompute the missing coordinate. However, since $\tilde{f}_n$ has relatively large degree, this would identify more points than desired. 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 modified approach to the problem in the next section.

4 An optimal representation

As the Semaev polynomials are symmetric in nature, they can be written in terms of the symmetric functions. We write

$$f_n(z_1, \ldots, z_n) = g_n(e_1(z_1, \ldots, z_n), \ldots, e_n(z_1, \ldots, z_n)), \tag{3}$$

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 (e.g. 2 instead of 4 for $n = 3$, and 8 instead of 32 for $n = 5$) 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 [29] for how to efficiently compute the symmetrized Semaev polynomials. In this sense,

$$g_n(s_1, \ldots, s_n) = 0 \tag{4}$$

also describes the points of T_n via the relations

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

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), as well as the polynomials $\tilde{e_i}$ that we obtain by Weil restriction from the symmetric functions in the q-powers of x:

$$s_i = \tilde{e}_i(x_0, \ldots, x_{n-1}),\ i = 1, \ldots, n. \tag{5}$$

Furthermore, we get exactly one new relation per equation (reducing modulo $x_i^q - x_i$ and applying Proposition 3, as before). Hence we have a total of n equations in the Weil restriction coordinates describing the symmetric functions. The qth powers in the exponents disappear thanks to the reduction, and each $\tilde{e}_i$ is homogeneous of degree i. A combination of the equations (4) and (5) enables us to give a compact representation of the affine points of $T_n = V(\mathbb{F}_q)$. It can be computed with the *compression* algorithm, the full point can be recovered (up to some small ambiguity) with the *decompression* algorithm.

Compression.

INPUT: $P = (X_0, \ldots, X_{n-1}, Y_0, \ldots, Y_{n-1}) \in V(\mathbb{F}_q)$

Compute the symmetric functions of the Frobenius conjugates of X:

$$S_i = \tilde{e}_i(X_0, \ldots, X_{n-1}),\ i = 1, \ldots, n-1$$

OUTPUT: $(S_1, \ldots, S_{n-1}) \in \mathbb{F}_q^{n-1}$

Decompression.

INPUT: $(S_1, \ldots, S_{n-1}) \in \mathbb{F}_q^{n-1}$

Solve $g_n(S_1, \ldots, S_{n-1}, t) = 0$ for t.
For each solution τ, find a solution (if it exists) of the system

$$\begin{aligned} S_1 &= \tilde{e}_1(x_0, \ldots, x_{n-1}) \\ &\vdots \\ S_{n-1} &= \tilde{e}_{n-1}(x_0, \ldots, x_{n-1}) \\ \tau &= \tilde{e}_n(x_0, \ldots, x_{n-1}). \end{aligned} \tag{6}$$

For the found solution $(X_0^{(j)}, \ldots, X_{n-1}^{(j)})$, recompute one of the y-coordinates $Y^{(j)}$ belonging to $X^{(j)} = X_0^{(j)} + \ldots + X_{n-1}^{(j)} \zeta^{n-1}$ using the curve equation.
If $(X^{(j)}, Y^{(j)}) \in T_n$, then add $\pm P = (X^{(j)}, \pm Y^{(j)})$ and all their Frobenius conjugates to the set of output points.

OUTPUT: All points of $T_n = V(\mathbb{F}_q)$ that have $(S_1, \ldots, S_{n-1})$ as compact representation

Remark 6. Because of Lemma 1, in the last step of the decompression algorithm, for each root $X^{(j)}$ of the polynomial f_n one needs to check that the point $(X^{(j)}, Y^{(j)}) \in T_n$. This step can in practice be eliminated for $n = 3, 5$, as discussed in Remark 5.

For a small set of points, equation (4) vanishes when evaluated in the given $S_1, \ldots, S_{n-1}$. For such points P, any $t \in \mathbb{F}_q$ solves the equation $g_n(S_1, \ldots, S_{n-1}, t) = 0$, 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 $P \in V(\mathbb{F}_q)$ the polynomial $g_n(S_1, \ldots, S_{n-1}, t)$ 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. Since the points of $V(\mathbb{F}_q)$ are described by $g_n(\tilde{e}_1(x_0, \ldots, x_{n-1}), \ldots, \tilde{e}_n(x_0, \ldots, x_{n-1}))$, we have $P \in$ Decompress(Compress(P)). The relevant question is how many more points the output may contain.

First of all, by compressing a point, we lose the ability to distinguish between Frobenius conjugates of points, since for each solution of system (6), 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 points and their negatives. 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. Notice also that it is enough to compute one solution of system (6), 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 1.

Notice moreover that there may be solutions τ of $g_n(S_1, \ldots, S_{n-1}, t) = 0$ for which system (6) has no solutions, and that not all the solutions of system (6) produce an equivalence class of points on the trace zero variety. E.g., 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 5 and 6, respectively. There, we also discuss the efficiency of our decompression algorithm and how it compares to the approaches of [35, 42].

Remark 7. 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 [15] and studied in detail by Lange [31, 32] and subsequently by Avanzi and Cesena [1].

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-Lin show in [16] 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:

1. Use the Montgomery ladder, which computes the x-coordinate of kP from X only.
2. 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.

All 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. [4, 34, 36, 14]. 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. This is the basis for our suggestion to follow the second approach when working with compressed points of T_n.

5 Explicit equations for extension degree 3

We give explicit equations for $n = 3$, where we write $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3 - \mu)$ and use $1, \zeta, \zeta^2$ as a basis for $\mathbb{F}_{q^3}|\mathbb{F}_q$. For completeness, we start with the standard equations for the trace zero variety (see [15]), although we do not make further use of them in our approach. They describe an open affine part of the trace zero variety (i.e. they hold when $x_1, x_2 \neq 0$):

$$\begin{aligned} y_0^2 + 2\mu y_1 y_2 &= x_0^3 + \mu x_1^3 + \mu^2 x_2^3 + 6\mu x_0 x_1 x_2 + A x_0 + B \\ 2y_0 y_1 + \mu y_2^2 &= 3x_0^2 x_1 + 3\mu x_0 x_2^2 + 3\mu x_1^2 x_2 + A x_1 \\ 2y_0 y_2 + y_1^2 &= 3x_0^2 x_2 + 3x_0 x_1^2 + 3\mu x_1 x_2^2 + A x_2 \\ x_1 y_2 &= x_2 y_1. \end{aligned} \tag{7}$$

The equation that we found in Section 3 only involves the x-coordinate and is

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

For Weil restriction, we write $x = x_0 + x_1\zeta + x_2\zeta^2$ and get

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

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. This gives

$$\begin{aligned} \tilde{f}_3(x_0, x_1, x_2) &= -3x_0^4 - 12\mu^2 x_0 x_2^3 - 12\mu x_0 x_1^3 + 18\mu x_0^2 x_1 x_2 \\ &\quad +9\mu^2 x_1^2 x_2^2 - 6Ax_0^2 + 6A\mu x_1 x_2 - 12Bx_0 + A^2. \end{aligned} \tag{8}$$

The symmetrized third Semaev polynomial is

$$g_3(s_1, s_2, s_3) = s_2^2 - 4s_1 s_3 - 4Bs_1 - 2As_2 + A^2 \tag{9}$$

and describes the trace zero subgroup via

$$\begin{aligned} s_1 &= x + x^q + x^{q^2} &&= 3x_0 \\ s_2 &= x^{1+q} + x^{1+q^2} + x^{q+q^2} &&= 3x_0^2 - 3\mu x_1 x_2 \\ s_3 &= x^{1+q+q^2} &&= x_0^3 - 3\mu x_0 x_1 x_2 + \mu x_1^3 + \mu^2 x_2^3. \end{aligned} \tag{10}$$

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

$$(s_1, s_2) = (3x_0, 3x_0^2 - 3\mu x_1 x_2),$$

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 (9). 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 moreover that the points $(0, s_2, s_3)$ with $s_2^2 - 2As_2 + A^2 = 0$ satisfy equation (9) for every s_3. The only ambiguity in decompression comes from solving system (10), 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

$$t_1 = x_0,\ t_2 = x_1 x_2,\ t_3 = x_1^3 + \mu x_2^3, \tag{11}$$

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

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

In this case, we recover t_3 from the equation

$$-3t_1^4 + 18\mu t_1^2 t_2 + 9\mu^2 t_2^2 - 12\mu t_1 t_3 - 12Bt_1 - 6At_1^2 + 6A\mu t_2 + A^2.$$

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 1, where 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.

Table 1: Average time in milliseconds for compression/decompression of one point when $n = 3$

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

All computations were done with Magma version 2.19.3 [7], running on one core of an Intel Xeon Processor X7550 (2.00 GHz) on a Fujitsu Primergy RX900S1. Our Magma programs are straight forward implementations of the methods presented here and are only meant as an indication. No particular effort has been put into optimizing them.

We give a concrete example, before concluding the section with a more detailed analysis of the efficiency of our algorithms.

Example 2. 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 (to save some space, we write only x-coordinates)

$$P = 260970034280824124824722 + 431820813779055023676698\zeta + 496444425404915392572065\zeta^2 \in T_3$$

and compute

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

where the results of decompression are exactly the Frobenius conjugates of P. In our Magma implementation, we solve system (10) over $\mathbb{F}_q$ similarly to how one would do it by hand, as described below. Note 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 the system directly is faster in all instances.

When using the second variant of the representation, we compute

$$(t_1, t_2) = (260970034280824124824722, 492721032528256431308437)$$

and naturally get the same result for decompression by solving system (11) in a similar way.

Operation count for representation in the s_i. Where possible, we count squarings (S), multiplications (M), and divisions (D) in $\mathbb{F}_q$. We do not count multiplication by constants, since they can often be chosen small (see [32]), and multiplication can then be performed by repeated addition. 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 1M+1D.
- Given s_1, s_2, s_3, we need to solve system (10) 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 [22]). 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 $27\mu^3 x_1^3$. In this way, we obtain the equation

$$27\mu^4 x_1^6 + 27\mu^3(x_0(s_2 - 2x_0^2) - s_3)x_1^3 + \mu^2(3x_0^2 - s_2)^3,$$

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 1D for the first, and a multiplication by the inverse of a cube root of unity for the other two values. Altogether, solving system (10) takes a total of at most 3S+2M+1D, 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+3M+2D, 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+1M+1D.
- To solve system (11), again assuming $x_1 \neq 0$, we have to find the roots of the equation

$$x_1^6 - t_3 x_1^3 + \mu t_2^3.$$

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 1D. Thus, solving the system takes a total of at most 2S+1M+1D, 1 square root, and 2 cube roots.

In total, decompression takes at most 4S+2M+2D, 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 1).

Comparison with Silverberg's method. The representation of [42] 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 proposes to represent 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. We disregard this step when estimating the complexity.

Since [42] 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) only. 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 $kP = \varphi^i(kQ)$ 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 [35] studies trace zero varieties for $n = 3$. He does not give explicit compression and decompression algorithms, but he derives an equation for the trace zero subgroup that may be used for such. In fact, his equation is identical to our equation (8), the Weil restriction of the (unsymmetrized) Semaev polynomial. However, he obtains it in a different way, namely, by eliminating from system (7).

Naumann suggests a compression method analogous to the one of Silverberg: A point is represented via the coordinates (x_1, x_2, ν, λ). For decompression, x_0 is recomputed from a quartic equation, $0 \leq \nu \leq 3$ determines which root of the equation is the correct x_0, and $0 \leq \lambda \leq 1$ determines the sign of the y-coordinate. Hence the quartic equation must be solved during both compression and decompression. Naumann's equation is different from Silverberg's, 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.

Security issues. To the extent of our knowledge, there are no known attacks on the DLP in T_3 whose complexity is lower than generic (square root) attacks, provided that one chooses

the parameters 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 index calculus methods, as detailed in [20] among many other works, do not yield an attack which is better than generic (square root) attacks in this setting, since the trace zero variety has dimension 2.

6 Explicit equations for extension degree 5

The fifth Semaev polynomial is too big to be printed here, but a computer program can easily work with it. It has total degree 32 and degree 8 in each indeterminate. The symmetrized fifth Semaev polynomial has total degree 8 and degree 6 in the last indeterminate. In fact, 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. It has a small number of terms compared to the original polynomial, but printing it here would still take several pages.

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 [42], 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

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

The compression algorithm computes $s_1, \ldots, s_4$ 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 $x_0, \ldots, x_4$ by solving the above 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 2. 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.

Table 2: Average time in milliseconds for compression/decompression of one point when $n = 5$

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

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

$$\begin{aligned} t_1 &= x_0 \\ t_2 &= x_1x_4 + x_2x_3 \\ t_3 &= x_1^2x_3 + x_1x_2^2 + \mu x_3^2x_4 + \mu x_2x_4^2 \\ t_4 &= \mu x_2^2x_3^2 + \mu x_1^2x_4^2 - \mu x_1x_3^3 - x_1^3x_2 - \mu x_2^3x_4 - \mu^2x_3x_4^3 + \mu x_1x_2x_3x_4 \\ t_5 &= x_1^5 + \mu x_2^5 + \mu^2x_3^5 + \mu^3x_4^5 + 5\mu x_1^2x_2x_3^2 + 5\mu x_1^2x_2^2x_4 + 5\mu^2x_2^2x_3x_4^2 \\ &\quad +5\mu^2x_1x_3^2x_4^2 - 5\mu x_1^3x_3x_4 - 5\mu^2x_2x_3^3x_4 - 5\mu^2x_1x_2x_4^3 - 5\mu x_1x_2^3x_3 \end{aligned} \tag{12}$$

and

$$\begin{aligned} s_1 &= 5t_1 \\ s_2 &= 10t_1^2 - 5\mu t_2 \\ s_3 &= 10t_1^3 - 15\mu t_1t_2 + 5\mu t_3 \\ s_4 &= 5t_1^4 - 15\mu t_1^2t_2 + 10\mu t_1t_3 + 5\mu t_4 \\ s_5 &= t_1^5 - 5\mu t_1^3t_2 + 5\mu t_1^2t_3 + 5\mu t_1t_4 + \mu t_5. \end{aligned} \tag{13}$$

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 (13) into the symmetrized fifth Semaev polynomial. Afterwards we may either recompute $s_1, \ldots, s_5$ from $t_1, \ldots, t_5$ according to system (13) 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 (12) directly for $x_0, \ldots, x_4$ 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 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 (6), 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 3. Let E be the curve $y^2 = x^3 + x + 135$ over $\mathbb{F}_q$, where $q = 2^{60} - 695$ is a 60-bit prime, and $\mu = 3$. The trace zero subgroup of $E(\mathbb{F}_{q^5})$ has prime order of 240 bits. We choose a

random point

$$\begin{aligned} P &= 697340666673436518 + 801324486821916366\zeta + 191523769921581598\zeta^2 \\ &\quad +193574581008452232\zeta^3 + 808272437423069772\zeta^4 \in T_5 \end{aligned}$$

and compute

$$\begin{aligned} &\text{Compress}(P) = (27938819546643747, 599177118073319826, 587362643323803394, 899440023033601132) \\ &\text{Decompress}(27938819546643747, 599177118073319826, 587362643323803394, 899440023033601132) \\ &\quad = \{697340666673436518 + 801324486821916366\zeta + 191523769921581598\zeta^2 \\ &\qquad + 193574581008452232\zeta^3 + 808272437423069772\zeta^4, \\ &\qquad 697340666673436518 + 836712212802745328\zeta + 506907366758395901\zeta^2 \\ &\qquad + 517000572714098077\zeta^3 + 268866625974497959\zeta^4, \\ &\qquad 697340666673436518 + 960543166171367987\zeta + 126552294958642222\zeta^2 \\ &\qquad + 448251978051599093\zeta^3 + 74315924307841334\zeta^4, \\ &\qquad 697340666673436518 + 810370833605859760\zeta + 539948230971075773\zeta^2 \\ &\qquad + 1032750511909194579\zeta^3 + 944608723064092684\zeta^4, \\ &\qquad 697340666673436518 + 49813814418649402\zeta + 940911346603997068\zeta^2 \\ &\qquad + 1142653655303485 81\zeta^3 + 209779298444190813\zeta^4\}. \end{aligned}$$

When using the second variant of the representation, we compute

$$(t_1, t_2, t_3, t_4) = (697340666673436518, 553115374027544004, 315951679773440541, 285024754797056479).$$

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. We give an example of a point for which three classes of Frobenius conjugates are recovered:

$$\begin{aligned} P &= 7600109093424145 70 + 5680645350588258 84\zeta + 244006548504894796\zeta^2 \\ &\quad +446522043528586762\zeta^3 + 731314735984238952\zeta^4 \in T_5. \end{aligned}$$

Operation count for representation in the s_i. Given $x_0, \ldots, x_4$, the numbers $t_1, \ldots, t_4$ can be computed with a total of 5S+13M according to (12). Then $s_1, \ldots, s_4$ can be computed from those numbers with 2S+3M as given in (13). This seems to be the best way to compute $s_1, \ldots, s_4$, 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 $s_1, \ldots, s_5$ from $t_1, \ldots, t_5$ and factor the polynomial, in which case this approach is exactly the same as the above. Or else we can solve system (12) 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 [42] for the case where the ground field has characteristic 3. The most costly parts of the compression and decompression 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.

Security issues. We briefly discuss the security issues connected with use of T_5 in DL-based and pairing-based cryptosystems.

Since T_5 is a group of size q^4, generic algorithms that solve the DLP in T_5 have complexity $O(q^2)$. Security threats in the context of DL-based cryptosystems are posed by algorithms for solving the DLP that achieve lower complexity. There are two types of algorithms that one needs to consider: First, cover attacks aim to transfer the DLP in $E(\mathbb{F}_{q^5})$ to the DLP in the Picard group of a curve of larger (but still rather low) genus, see [21, 9]. The DLP is then solved there using index calculus methods. Combining the results of [9] and [10], it is sometimes possible to map the DLP from T_5 into the Picard group of a genus 5 curve (which is usually not hyperelliptic), where it can be solved with probabilistic complexity $\tilde{O}(q^{4/3})$ following the approach of [11]. However, only a very small proportion of curves is affected by this attack, and such curves should be avoided in practice. Moreover, in order to avoid isogeny attacks, the curve should be chosen such that 4 does not divide the order of T_5, see [9]. Second, the index calculus attack of [20] applies to T_5 and has complexity $\tilde{O}(q^{3/2})$. This makes T_5 not an ideal group to use in a DL-based cryptosystem. Notice that in practice, however, the constant in the O is very large, since the attack requires Gröbner basis computations, which are very time consuming (their worst case complexity is doubly exponential in the size of the input), and often do not terminate in practice. It is our impression that more in depth study is needed in order to give a precise estimate of the feasibility of such an attack for a practical choice of the parameters. We carried on preliminary experiments, which indicate that a straightforward application of the method from [20] to T_5 yields a system of equations which is very costly to compute (it requires computing the Weil descent of the fifth Semaev polynomial) and which Magma cannot solve in several weeks and using more than 300 GB of memory on the same machine that we used to carry out the experiments reported on in Sections 5 and 6 of this article. Notice that solving such a system would (possibly) produce one relation, to be then used in an index calculus attack. Therefore, in practice one would need to solve many such systems, in order to produce the relations needed for the linear algebra step of the index calculus attack.

Trace zero varieties are even more interesting in the context of pairing-based cryptography. The main motivation comes from [40], where Rubin and Silverberg show that supersingular abelian varieties of dimension greater than one offer more security than supersingular elliptic curves, for the same group size. Trace zero varieties are explicitly mentioned in [40] as one of the most relevant examples of abelian varieties for pairing-based cryptography. 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 [40] hold over fields of any characteristic, one should avoid fields of small characteristic, so that the recent attacks from [23, 28, 24, 2, 3] do not apply. Over a field of large characteristic, the cover and index calculus attacks that we discussed in the previous paragraph 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 have lower complexity than cover and index calculus attacks in the lines of [20, 21, 9, 11]. In some cases however, the choice of the security parameter may need to be adjusted, according to the complexity of these index calculus attacks. As an example, let us discuss the choice of parameters for a pairing with

80-bits 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 ends up in 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 an 80-bit security, so the attacks from [20, 21, 9, 11] end 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 achieve 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, the complexity of solving a DLP in T_5 and in $\mathbb{F}_{q^{20}}$ are both about 80-bits when $q \sim 2^{53}$. Summarizing, the complexity of the DLP in T_5 coming from the works [20, 21, 9, 11] 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 on, 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.

7 Conclusion

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 in size for $n = 3$ and for $n = 5$. Our polynomials have lower degree than those used in the representations of [42] (1 compared to 4 for $n = 3$, and 6 compared to 27 for $n = 5$) and [35] (1 compared to 3 for $n = 3$), thus allowing more efficient compression and 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.

Acknowledgements We thank Pierrick Gaudry and Peter Schwabe for helpful discussions and Tanja Lange for pointing out the work of Naumann. We are grateful to the mathematics department of the Univerity of Zürich for access to their computing facilities. The authors were supported by the Swiss National Science Foundation under Grant No. 123393.

References

[1] 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.

[2] R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, and P. Zimmermann. Discrete logarithm in GF(2^{809}) with FFS. Available at `http://hal.inria.fr/hal-00818124/`.

[3] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. Available at http://arxiv.org/abs/1306.4244, 2013.

[4] 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.

[5] G. Blady. Die Weil-Restriktion elliptischer Kurven in der Kryptographie. Master's thesis, Univerität GHS Essen, 2002.

[6] J. W. Bos, C. Costello, H. Hisil, and K. Lauter. High-performance scalar multiplication using 8-dimensional GLV/GLS decomposition. Available at http://eprint.iacr.org/2013/146, accepted at CHES '13, 2013.

[7] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24:235–265, 1997.

[8] 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.

[9] C. Diem. The GHS attack in odd characteristic. *Ramanujan Math. Soc.*, 18(1):1–32, 2003.

[10] 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, Berlin–Heidelberg–New York, 2006. Springer.

[11] C. Diem and S. Kochinke. Computing discrete logarithms with special linear systems. Available at http://www.math.uni-leipzig.de/~diem/preprints/dlp-linear-systems.pdf, 2013.

[12] C. Diem and J. Scholten. An attack on a trace-zero cryptosystem. Available at http://www.math.uni-leipzig.de/diem/preprints.

[13] P. N. J. Eagle, S. D. Galbraith, and J. Ong. Point compression for Koblitz curves. *Adv. Math. Commun.*, 5(1):1–10, 2011.

[14] 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.

[15] 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.

[16] S. D. Galbraith and X. Lin. Computing pairings using x-coordinates only. *Des. Codes Crytogr.*, 50(3):305–324, 2009.

[17] 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.

[18] S. D. Galbraith and B. A. Smith. Discrete logarithms in generalized Jacobians. Available at http://uk.arxiv.org/abs/math.NT/0610073, 2006.

[19] 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.

[20] 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.

[21] P. Gaudry, F. Hess, and N.P. Smart. Constructive and destructive facets of Weil descent. *J. Cryptology*, 15(1):19–46, 2002.

[22] J. Gerhard and J. von zur Gathen. *Modern Computer Algebra.* Cambridge University Press, Cambridge, 1999.

[23] 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 $\mathbb{F}_{2^{1971}}$. Available at http://eprint.iacr.org/2013/074, 2013.

[24] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel. Solving a 6120-bit DLP on a desktop computer. Available at http://eprint.iacr.org/2013/306, 2013.

[25] G. Gong and L. Harn. Public-key cryptosystems based on cubic finite field extensions. *IEEE Trans. Inform. Theory*, 45(7):2601–2605, 1999.

[26] 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.

[27] 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.

[28] A. Joux. A new index calculus algorithm with complexity $L(1/4+o(1))$ in very small characteristic. Available at http://eprint.iacr.org/2013/095, 2013.

[29] 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 Journal of Cryptology, Springer, DOI: 10.1007/s00145-011-9116-z, 2012.

[30] 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.

[31] T. Lange. *Efficient Arithmetic on Hyperelliptic Curves*. PhD thesis, Univerität GHS Essen, Available at http://www.hyperelliptic.org/tanja/preprints.html, 2001.

[32] T. Lange. Trace zero subvarieties of genus 2 curves for cryptosystem. *Ramanujan Math. Soc.*, 19(1):15–33, 2004.

[33] 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.

[34] 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.

[35] N. Naumann. Weil-Restriktion abelscher Varietäten. Master's thesis, Univerität GHS Essen, Available at http://web.iem.uni-due.de/ag/numbertheory/dissertationen, 1999.

[36] T. Oliveira, J. López, D. F. Aranha, and F. Rodríguez-Henríquez. Lambda coordinates for binary elliptic curves. Available at http://eprint.iacr.org/2013/131, accepted at CHES '13, 2013.

[37] 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.

[38] 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.

[39] K. Rubin and A. Silverberg. Using primitive subgroups to do more with fewer bits. In D. Buell, editor, *Algorithmic Number Theory (ANTS VI)*, volume 3076 of *LNCS*, pages 18–41, Berlin–Heidelberg–New York, 2004. Springer.

[40] K. Rubin and A. Silverberg. Using abelian varieties to improve pairing-based cryptography. *J. Cryptology*, 22(3):330–364, 2009.

[41] I. Semaev. Summation polynomials of the discrete logarithm problem on elliptic curves. Available at http://eprint.iacr.org/2004/031, 2004.

[42] A. Silverberg. Compression for trace zero subgroups of elliptic curves. *Trends Math.*, 8:93–100, 2005.

[43] 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.

[44] 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.