# AN OPTIMAL REPRESENTATION FOR THE TRACE ZERO SUBGROUP

ELISA GORLA AND MAIKE MASSIERER

ABSTRACT. We give an optimal-size representation for the elements of the trace zero subgroup of the Picard group of an elliptic or hyperelliptic curve of any genus, with respect to a field extension of any prime degree. The representation is via the coefficients of a rational function, and it is compatible with scalar multiplication of points. We provide efficient compression and decompression algorithms, and complement them with implementation results. We discuss in detail the practically relevant cases of small genus and extension degree, and compare with the other known compression methods.

## 1. INTRODUCTION

Public key cryptography provides methods for secure digital communication. Among all public key cryptosystems, a relevant role is played by those based on the discrete logarithm problem (DLP). Such cryptographic systems work in finite groups which must satisfy three basic requirements: Computing the group operation must be efficient, the DLP must be hard, and there must be a convenient and compact representation for the elements.

One such group is the trace zero subgroup of the Picard group of an elliptic or hyperelliptic curve. 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. While it has long been established that the trace zero subgroup provides efficient arithmetic and good security properties, an efficient representation was only known for special parameters. We bridge this gap by proposing an optimal-size representation for the elements of trace zero subgroups associated to elliptic curves and hyperelliptic curves of any genus, with respect to field extensions of any prime extension degree.

The trace zero subgroup can be realized as the $\mathbb{F}_q$-rational points of the trace zero variety, an abelian variety built by Weil restriction from the original curve. It was first proposed in the context of cryptography by Frey [Fre99] and further studied by Naumann [Nau99], Weimerskirch [Wei01], Blady [Bla02], Lange [Lan01, Lan04], Silverberg [Sil05], Avanzi–Cesena [AC07], Cesena [Ces08, Ces10], and Diem–Scholten [DS], among others. 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 solving the DLP in the Jacobian can be reduced to solving the DLP in the trace zero subgroup. Therefore, trace zero 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 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 examples of groups with largest known security parameter come from trace zero subgroups.

Scalar multiplication in the trace zero subgroup is particularly efficient, due to a speed-up using the Frobenius endomorphism, see [Lan01, Lan04, AC07]. This technique is similar to the one used on Koblitz curves [Kob91] and has been afterwards applied to GLV/GLS curves [GLV01, GLS11], which are the basis for several recent implementation speed records for elliptic curve arithmetic [LS12, FHLS14, BCHL13]. In [AC07], Avanzi and Cesena show that trace zero subgroups often deliver better scalar multiplication performance than elliptic curves. E.g., scalar multiplication in trace zero subgroups of elliptic curves over a degree 5 extension field is almost 3 times faster than in 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 arithmetic performance, even though such systems sacrifice some memory and bandwidth.

In this paper, we solve this problem by providing a representation for the elements of trace zero subgroups which is both efficiently computable and optimal in size. 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 solution would be representing an element of the trace zero subgroup via $(n-1)g$ elements of $\mathbb{F}_q$. Such representations have been proposed by Naumann [Nau99, Chapter 4.2] for trace zero subgroups of elliptic curves and by Lange [Lan04] for trace zero varieties associated to hyperelliptic curves of genus 2, both with respect to cubic field extensions, and by Silverberg [Sil05] and Gorla–Massierer [GM15] for 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].

In this paper we give 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, 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 a DLP-based primitive, 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.

We also give a compression algorithm to compute the representation, and a decompression algorithm to recover the original divisor class. We show that our algorithms are comparable with or more efficient than all previously known methods, when one compares the total time required for compression and decompression.

The paper is organized as follows: In Section 2 we give some preliminaries on (hyper)elliptic curves, the trace zero variety, and optimal representations. In Section 3 we discuss the representation, together with compression and decompression algorithms, and we specialize these results to elliptic curves in Section 4. In Section 5 we present some implementation results, as well as a detailed comparison with the other compression methods. Finally, in the Appendix we give explicit equations for the relevant cases $g = 1, n = 3, 5$ and $g = 2, n = 3$.

## 2. Preliminaries

We start by recalling the definitions and basic facts that we will need in this paper, and fixing some notation.

### 2.1. Elliptic and hyperelliptic curves.

Let $C$ be a projective elliptic or hyperelliptic curve of genus $g$ defined over a finite field $\mathbb{F}_q$ that has an $\mathbb{F}_q$-rational Weierstraß point. For ease of exposition, we assume that $\mathbb{F}_q$ does not have characteristic 2. By making the necessary adjustments, the content of this paper carries over to the binary case. If $\mathbb{F}_q$ has odd characteristic, then $C$ can be given by an affine equation of the form

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

with $f \in \mathbb{F}_q[x]$ monic of degree $2g + 1$ and with no multiple zeros. We denote by $\mathcal{O}$ the point at infinity and by $\mathrm{Div}_C$ the group of divisors on $C$. Let $w$ be the involution

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

The Frobenius map on $C$ is defined as

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

Both $w$ and $\varphi$ extend to group homomorphisms on $\mathrm{Div}_C$.

Let $\mathbb{F}_{q^n}$ be an extension field of $\mathbb{F}_q$, $n \geq 1$. A divisor $D$ is $\mathbb{F}_{q^n}$-*rational* if $\varphi^n(D) = D$. We denote by $\mathrm{Div}_C(\mathbb{F}_{q^n})$ the $\mathbb{F}_{q^n}$-rational divisors on $C$. $\mathrm{Div}_C(\mathbb{F}_{q^n})$ is a subgroup of $\mathrm{Div}_C$.

Let $D_1 = a_1 P_1 + \ldots + a_k P_k - a\mathcal{O}, D_2 = b_1 P_1 + \ldots + b_k P_k - b\mathcal{O} \in \mathrm{Div}_C$, $a_i, b_i, a, b \in \mathbb{N} \cup \{0\}$, be two divisors of degree zero. If $a_i \leq b_i$ for all $i$ we write $D_1 \leq D_2$.

As usual in the cryptographic setting, we work in the Picard group $\mathrm{Pic}_C^0$ of $C$. This is the group of degree zero divisor classes, modulo principal divisors. For any $D, D_1, D_2 \in \mathrm{Div}_C$, we write $[D]$ for the equivalence class of $D$ in $\mathrm{Pic}_C^0$ and $D_1 \sim D_2$ for $[D_1] = [D_2]$. The $\mathbb{F}_{q^n}$-rational divisor class $[D]$ is the equivalence class of the $\mathbb{F}_{q^n}$-rational divisor $D$. The subgroup of $\mathrm{Pic}_C^0$ consisting of the $\mathbb{F}_{q^n}$-rational divisor classes is denoted by $\mathrm{Pic}_C^0(\mathbb{F}_{q^n})$.

A divisor $D = P_1 + \ldots + P_r - r\mathcal{O} \in \mathrm{Div}_C^0$ is *semi-reduced* if $P_i \in C \setminus \{\mathcal{O}\}$ and $P_i \neq w(P_j)$ for $i \neq j$. $D$ is *reduced* if it is semi-reduced and in addition $r \in \{0, \ldots, g\}$. Notice that $D$ is reduced with $r = 0$ if and only if $[D] = 0$.

It follows from the Riemann–Roch Theorem that every degree zero divisor class can be represented by a unique reduced divisor. For any divisors $D_1, D_2 \in \mathrm{Div}_C^0$, we denote by $D_1 \oplus D_2$ the reduced divisor such that $[D_1 \oplus D_2] = [D_1 + D_2]$. When $C$ is an elliptic curve, then each non-zero element of $\mathrm{Pic}_C^0$ is uniquely represented by a divisor of the form $P - \mathcal{O}$ with $P \in C$. In fact, we have $C \cong \mathrm{Pic}_C^0$ as groups via $P \mapsto [P - \mathcal{O}]$. For elliptic curves, we denote a divisor class by the unique corresponding $P \in C$. In particular, we denote $0 \in \mathrm{Pic}_C^0$ by the point $\mathcal{O}$.

There is a one-to-one correspondence between semi-reduced divisors $D = P_1 + \ldots + P_r - r\mathcal{O}$ and pairs of polynomials $(u, v)$ such that $u$ is monic, $\deg v < \deg u$, and $u \mid v^2 - f$: Given a 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 polynomials $u, v$ as above, let $D = \Delta - \deg(\Delta)\mathcal{O}$ where $\Delta$ is the effective divisor with defining ideal $I_\Delta = (u(x), y - v(x))$. It is easy to show that $D$ is semi-reduced. Notice that since $u \mid v^2 - f$, then $y^2 - f \in (u, y - v)$. The correspondence restricts to a correspondence between reduced divisors and pairs of polynomials $(u, v)$ such that $u$ is monic, $\deg v < \deg u \leq g$, and $u \mid v^2 - f$.

A commonly used representation for divisor classes is the *Mumford representation*. An element $[D] \in \mathrm{Pic}_C^0$ with $D$ a reduced divisor is represented by the pair of polynomials $[u(x), v(x)]$ associated to it in the correspondence described in the previous paragraph. The Mumford representation is particularly useful when computing with divisor classes, and all algorithms given

in this paper make use of this representation. If $C$ is an elliptic curve, then the Mumford representation of $P = (X, Y) \in C$ is $[x - X, Y]$. It follows from the definition that the Mumford representation of $[0]$ is $[1, 0]$. A convenient property of the Mumford representation is that $\mathbb{F}_{q^n}$-rationality of divisor classes is easily detected: $[u, v] \in \mathrm{Pic}_C^0(\mathbb{F}_{q^n})$ if and only if $u, v \in \mathbb{F}_{q^n}[x]$.

By definition, a reduced divisor $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ with $[D] = [u, v]$ is *prime* if $u \in \mathbb{F}_{q^n}[x]$ is an irreducible polynomial. This is equivalent to the statement that $(u, y - v)$ is a prime ideal of $\mathbb{F}_{q^n}[x, y]/(y^2 - f(x))$. Notice that being prime depends on the choice of $\mathbb{F}_{q^n}$. Sometimes we write a divisor as a sum of prime divisors: $D = D_1 + \ldots + D_t$, with $D_i \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ prime. The prime divisors $D_1, \ldots, D_t$ are unique up to permutation, but not necessarily distinct. If $[D_i] = [u_i, v_i]$ is the Mumford representation, then $u = \prod_{i=1}^t u_i$ is the irreducible factorization of $u \in \mathbb{F}_{q^n}[x]$.

Cantor's Algorithm performs the addition of divisor classes in the Mumford representation. For elliptic curves and hyperelliptic curves of genus 2, there exist explicit addition formulas that are easier to use and more efficient than Cantor's Algorithm (see [Was08] and [Lan05]).

2.2. **The trace zero variety and optimal representations.** The trace endomorphism in the divisor group of $C$ with respect to the extension $\mathbb{F}_{q^n} | \mathbb{F}_q$ is defined by

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

Throughout the paper, we denote by $u^\varphi$ 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 $u u^\varphi \cdots u^{\varphi^{n-1}}$ by $u^{1+\varphi+\ldots+\varphi^{n-1}}$ or by $N(u)$, and we call it the *norm* of $u$.

**Lemma 2.1.** *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.* (i) Let $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ be a prime divisor with $[D] = [u, v]$, $u \in \mathbb{F}_{q^n}[x]$ irreducible. Then $\mathrm{Tr}(D)$ has $u$-polynomial $N(u) = u u^\varphi \cdots u^{\varphi^{n-1}}$, where all the $u^{\varphi^j}$ 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 $u^{\varphi^j}$, 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) This is a restatement of the well known fact that that $u \in \mathbb{F}_{q^n}[x] \setminus \mathbb{F}_q[x]$ is irreducible if and only if $N(u) = u u^\varphi \cdots u^{\varphi^{n-1}} \in \mathbb{F}_q[x]$ is irreducible. $\square$

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

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

We are interested in the kernel of this map.

**Definition 2.2.** Let $n$ be a prime number. Then the *trace zero subgroup* of $\mathrm{Pic}_C^0(\mathbb{F}_{q^n})$ is

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

Using Weil restriction, the points of $T_n$ can be viewed as the $\mathbb{F}_q$-rational points of a $g(n-1)$-dimensional variety defined over $\mathbb{F}_q$, called the *trace zero variety*. For a proof and more details, see [ACD$^+$06, Chapters 7.4.2 and 15.3].

Interest in the trace zero variety in the cryptographic context was first raised by Frey in [Fre99]. The main advantages of working in $T_n$ are that addition in the trace zero subgroup may be sped up considerably by using the Frobenius endomorphism, and that it yields high security parameters in the context of pairing-based cryptography, for some values of $n$ and $g$. Moreover, the DLP in $\mathrm{Pic}_C^0(\mathbb{F}_{q^n})$ is as hard as the DLP in $T_n$. Therefore, working in $T_n$ allows

us to reduce the key length with respect to $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ without compromising the hardness of the DLP. In order to reduce the key length however, one needs to find an efficient representation for its elements. In this paper, we give an optimal one for any $g$ and any prime $n$.

We start by showing that solving the DLP in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ can be reduced to solving the DLP in $T_n$.

**Proposition 2.3.** *We have a short exact sequence*

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

*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. By the standard reduction obtained by combining an effective version of the Chinese Remainder Theorem and the Pohlig–Hellman Algorithm, we may assume without loss of generality that we are solving a DLP of the form $a[D] = [D']$, where $[D], [D'] \in \mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ and $[D]$ has prime order. If $[\varphi(D) - D] \neq 0$, then $[\varphi(D) - D]$ and $[D]$ have the same order, and the DLP may be mapped to $T_n$ via $[\varphi - \mathrm{id}]$ and solved there. Else, $[D] \in \mathrm{Pic}^0_C(\mathbb{F}_q)$. $\qquad\square$

**Remark 2.4.** We stress that the choice of good parameters is crucial for the security of trace zero cryptosystems. While Lange [Lan04], Avanzi–Cesena [AC07], and Rubin–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 example, Weil descent attacks (see [GHS02, Die03, DS]) and index calculus attacks (see [Gau09, EGT11, Die11]) may apply. However, Weil descent attacks only apply to a very small proportion of all curves, and index calculus attacks often have large constants hidden in the asymptotic complexity analysis, thus making them very hard to realize in practice. Nevertheless, special care must be taken to choose good parameters and avoid weak curves. E.g., for $g = 1$ and $n = 3$ and for most curves, computing a DLP in the trace zero subgroup has square root complexity.

**Remark 2.5.** As a consequence of the exact sequence in Proposition 2.3 we obtain that the cardinality of the trace zero subgroup may be computed easily in terms of the coefficients of the characteristic polynomial, see also [ACD+06, Chapter 15.3.1]. In particular, 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 $\mathbb{F}_{q^{n-1}}$.

The question of finding an optimal-size representation for the elements of the trace zero subgroup has been investigated in previous works both for elliptic and hyperelliptic curves, and it is stated as an open problem in the conclusions of [AC07]. The analogous problem for primitive subgroups of finite fields leads to torus-based cryptography.

**Definition 2.6.** A *representation of size* $\ell$ for the elements of a finite set $G$ is a map

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

with the property that an element of $\mathbb{F}_2^\ell$ has at most $d$ inverse images, where $d$ is a constant. Given $g \in G$, $x \in \mathrm{Im}\,\mathcal{R}$, we refer to computing $\mathcal{R}(g)$ as *compression* and $\mathcal{R}^{-1}(x)$ as *decompression*.

By taking $d = 1$ in the definition, one obtains the intuitive definition of a representation as an injective map. In this paper, we wish to consider representations which identify at most

$d$ elements of $G$, where $d$ is small. In our setup, any representation $\mathcal{R}$ induces an injective representation

$$\overline{\mathcal{R}} : G/\sim \longrightarrow \mathbb{F}_2^\ell,$$

where $g \sim h$ iff $\mathcal{R}(g) = \mathcal{R}(h)$ for any $g, h \in G$. We sometimes do not distinguish between $\mathcal{R}$ and $\overline{\mathcal{R}}$, and say that $x \in \operatorname{Im}\mathcal{R}$ is a representation for the *class* $\mathcal{R}^{-1}(x)$.

**Definition 2.7.** A representation $\mathcal{R} : G \longrightarrow \mathbb{F}_2^\ell$ of size $\ell$ for the elements of a finite set $G$ is *optimal* if

$$\ell = \lceil \log_2 |G| \rceil + O(1).$$

Intuitively, a representation is optimal if $\ell$ is the smallest possible length of a binary representation of the elements of $G$, up to an additive constant. Notice that, since $\log_2 |G| - \log_2 d = \log_2(|G|/d) \leq \log_2 |G/\sim| \leq \log_2 |G|$, one has that

$$\lceil \log_2 |G| \rceil = \lceil \log_2 |G/\sim| \rceil + O(1)$$

for any constant $d$. In particular, the length of an optimal representation for the elements of $G$ does not depend on $d$, and Definition 2.7 is well-posed.

Notice moreover that $\mathcal{R}$ is an optimal representation for the elements of $G$ if and only if $\overline{\mathcal{R}}$ is an optimal representation for the elements of $G/\sim$.

**Remark 2.8.** The problem of representing the elements of $\mathbb{F}_q$ via binary strings of length $\lceil \log_2 q \rceil$ is well studied. Therefore, a representation for a set $G$ may be given via $\mathcal{R} : G \longrightarrow \mathbb{F}_q^m$ or more generally via

$$\mathcal{R} : G \longrightarrow \mathbb{F}_q^m \times \mathbb{F}_2^k.$$

By definition, such a representation is optimal if

$$m\lceil \log_2 q \rceil + k = \lceil \log_2 |G| \rceil + O(1).$$

In particular, if $|G| = \Theta(q^m)$ and $k = O(1)$, then the representation is optimal, since $\lceil \log_2 |G| \rceil = \lceil \log_2 q^m \rceil + O(1) = m\lceil \log_2 q \rceil + k + O(1)$.

In this paper, we construct a representation of the form

$$\mathcal{R} : T_n \longrightarrow \mathbb{F}_q^{g(n-1)} \times \mathbb{F}_2$$

with the property that each element in the image has at most $n^g$ inverse images. Since $|T_n| = \Theta(q^{g(n-1)})$, then our representation is optimal by Remark 2.8.

**Example 2.9.** The usual representation for points on an elliptic curve $E$ defined over $\mathbb{F}_q$ is

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

This is an optimal representation of size $\lceil \log_2 q \rceil$. Optimality follows from Hasse's Theorem, which states that $|E(\mathbb{F}_q)| = \Theta(q)$.

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 $\mathbb{F}_q$. 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. This is compatible with scalar multiplication, since $\mathcal{R}(P) = \mathcal{R}(Q)$ if and only if $\mathcal{R}(kP) = \mathcal{R}(kQ)$, for any $P, Q \in E$ and for all $k \in \mathbb{N}$.

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

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

of size $\lceil \log_2 q \rceil + 1$, which is still optimal since $\lceil \log_2 q \rceil + 1 = \lceil \log_2 |E(\mathbb{F}_q)| \rceil + O(1)$ by Hasse's Theorem.

This logic can also be applied to higher genus hyperelliptic curves.

**Example 2.10.** Let $C$ be a hyperelliptic curve of genus $g$ defined over $\mathbb{F}_q$ and define

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

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 support of the reduced divisor $D = P_1 + \ldots + P_r - r\mathcal{O}$, but not about the signs of the corresponding $y$-coordinates. Therefore $\mathcal{R}$ identifies divisor classes of the form $\{[w^{i_1}(P_1) + w^{i_2}(P_2) + \ldots + w^{i_r}(P_r) - r\mathcal{O}] \mid i_j \in \{0, 1\}\}$, hence up to $2^g$ elements of $\mathrm{Pic}^0_C(\mathbb{F}_q)$. As before, one can use $g$ extra bits to store these signs, making the representation injective (see Hess–Seroussi–Smart [HSS01]). A different representation for the elements of $\mathrm{Pic}^0_C(\mathbb{F}_q)$ of size $g\lceil \log_2 q \rceil + g$ is given by Stahlke [Sta04].

**Remark 2.11.** Since $T_n \subset \mathrm{Pic}^0_C(\mathbb{F}_{q^n})$, we may use the representation of Example 2.10 for points of the trace zero subgroup. However this representation is not optimal for $T_n$, since it has size $ng\lceil \log_2 q \rceil + 1$, while $\lceil \log_2 |T_n| \rceil = (n-1)g\lceil \log_2 q \rceil + O(1)$.

## 3. AN OPTIMAL REPRESENTATION FOR THE TRACE ZERO SUBGROUP VIA RATIONAL FUNCTIONS

In this section, we give an optimal representation for the points of the trace zero subgroup. A simple example is the case of elliptic curves $E$ and extension degree $n = 2$, where

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

Hence the $x$-coordinate of the points of $T_2$ yields an optimal representation (see [GM15, Proposition 2]). This statement can be generalized to higher genus curves when $n = 2$.

**Proposition 3.1.** *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*

$$
(1) \qquad T_2 = \{[u, v] \in \mathrm{Pic}^0_C(\mathbb{F}_{q^2}) \mid u \in \mathbb{F}_q[x], v^\varphi = -v\}.
$$

*Therefore, the map*

$$
\begin{array}{rcl}
\mathcal{R} : \quad T_2 & \longrightarrow & \mathbb{F}_q^g \times \mathbb{F}_2 \\
[u, v] & \longmapsto & (u_0, \ldots, u_{g-1}, \delta)
\end{array}
$$

*yields an optimal representation for the elements of $T_2$, where $u = \sum_{i=0}^g u_i x^i$ is monic of degree $0 \leq r \leq g$, $\delta = 1$ if $\deg u = g$, and $\delta = 0$ otherwise.*

*Proof.* Let $D$ be a reduced divisor with Mumford representation $[u, v]$. Then: $[D] \in T_2$ if and only if $[D] = [w(\varphi(D))]$. Since $[w(\varphi(D))] = [u^\varphi, -v^\varphi]$ and both $D$ and $w(\varphi(D))$ are reduced, then $[D] = [w(\varphi(D))]$ if and only if $u = u^\varphi$ and $v = -v^\varphi$ if and only if $u \in \mathbb{F}_q[x]$ and $v^\varphi = -v$. This proves equality (1).

Equality (1) allows us to represent $[D] = [u, v] \in T_2$ via $u$, with $u \in \mathbb{F}_q[x]$ monic of degree $r$, $0 \leq r \leq g$. In turn, $u$ may be represented via its $g$ coefficients $u_0, \ldots, u_{g-1} \in \mathbb{F}_q$, together with the extra bit $\delta$ which indicates whether $\deg u = g$ or not. This representation identifies $[D]$ and its Frobenius conjugate $[w(D)] = [\varphi(D)]$. Finally, it has length $g\lceil \log_2 q \rceil + 1 = \lceil \log_2 |T_2| \rceil + O(1)$, hence it is optimal. $\qquad \square$

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 defined over $\mathbb{F}_q$ since $\mathrm{Tr}(D)$ is, and it 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 $\mathbb{F}_q$ plus an extra bit. This gives an optimal representation for the elements of $T_n$, which identifies at most $n^g$ divisor classes.

**Theorem 3.2.** *Let $D = P_1 + \ldots + P_r - r\mathcal{O}$ be a reduced divisor such that $[D] = [u, v] \in T_n$, and let $h_D \in \mathbb{F}_q(C)$ be a function such that $\mathrm{div}(h_D) = \mathrm{Tr}(D)$. 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 $N(u)$ denotes the norm of $u$ relative to $\mathbb{F}_{q^n} | \mathbb{F}_q$.*

(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) *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 $h_F = h_D$.*

*Proof.* Since $[D] \in T_n$, we have $0 \sim \mathrm{Tr}(D) \in \mathrm{Div}_C(\mathbb{F}_q)$. Hence there exists an $h_D \in \mathbb{F}_q(C)$ such that $\mathrm{div}(h_D) = \mathrm{Tr}(D)$. 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 $\mathcal{O}$. Modulo the curve equation $y^2 = f(x)$, the polynomial $h_D \in \mathbb{F}_q[x, y]$ has the desired shape.

(ii) By definition, $h_D$ has zeros $\varphi^j(P_1), \ldots, \varphi^j(P_r), j = 0, \ldots, n-1$, and pole $nr\mathcal{O}$. Therefore, $h_D \circ w = h_{D,1}(x) - y h_{D,2}(x)$ has zeros $w(\varphi^j(P_1)), \ldots, w(\varphi^j(P_r)), j = 0, \ldots, n-1$ and pole $nr\mathcal{O}$. Since $H_D(x) = h_D(h_D \circ w) \in \mathbb{F}_q[x, y]/(y^2 - f(x))$, 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 pole $2nr\mathcal{O}$. Therefore $H_D = N(u)$, up to multiplication by a constant.

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

(iv) 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 2.1 (ii). By Lemma 2.1 (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 at most $n^t \leq n^g$. $\square$

**Remark 3.3.** If $n = 2$ and $[D] = [u(x), v(x)] \in T_2$, then $h_D(x, y) = u(x)$. Hence Theorem 3.2 recovers the optimal representation from Proposition 3.1.

**Remark 3.4.** Let $D \in \mathrm{Div}_C^0(\mathbb{F}_{q^n})$ be a reduced divisor, $D = D_1 + \ldots + D_t$ with $D_i \in \mathrm{Div}_C^0(\mathbb{F}_{q^n})$ 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 $F = \varphi^{j_1}(P) + w(\varphi^{j_2}(P)) - 2\mathcal{O}$ 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 divisors classes $[F] \in T_n$ which have $\mathrm{Tr}(F) = \mathrm{Tr}(D)$, but $F$ is not a reduced divisor. In our decompression algorithm, for a given

$\alpha = \mathcal{R}([D])$ we recover one reduced $F \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ such that $\mathcal{R}([F]) = \alpha$. Such an $F$ uniquely identifies $\mathcal{R}^{-1}(\mathcal{R}([D]))$.

The following corollary clarifies how Theorem 3.2 gives an optimal representation for the elements of $T_n$, consisting of $(n-1)g$ elements of $\mathbb{F}_q$ and a bit. Using standard techniques, the representation may be made injective at the cost of appending $g\lfloor \log_2 n \rfloor + 1$ bits to it.

**Corollary 3.5.** *Let $n \geq 3$, let $0 \neq D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ be a reduced divisor of degree zero such that $[D] = [u, v] \in T_n$, and let $r = \deg u$. Set $d_1 = \lfloor \frac{ng}{2} \rfloor$ and $d_2 = \lfloor \frac{(n-2)g-1}{2} \rfloor$. Let $h_D = h_{D,1}(x) + yh_{D,2}(x) \in \mathbb{F}_q[x,y]$ be such that $\mathrm{div}(h_D) = \mathrm{Tr}(D)$, where $h_{D,1} = \gamma_{d_1}x^{d_1} + \ldots + \gamma_1 x + \gamma_0$, $h_{D,2} = \beta_{d_2}x^{d_2} + \ldots + \beta_1 x + \beta_0$. Let $h_{D,1}$ be monic if $r$ is even, and $h_{D,2}$ be monic if $r$ is odd. If $r = g$ let $\delta = 1$, else let $\delta = 0$. Define:*

- *If $g$ is even, then*

$$\begin{aligned}
\mathcal{R} : T_n &\longrightarrow \mathbb{F}_q^{(n-1)g} \times \mathbb{F}_2 \\
[D] &\longmapsto (\beta_0, \ldots, \beta_{d_2}, \gamma_0, \ldots, \gamma_{d_1 - 1}, \delta) \\
[0] &\longmapsto (0, \ldots, 0).
\end{aligned}$$

- *If $g$ is odd, then*

$$\begin{aligned}
\mathcal{R} : T_n &\longrightarrow \mathbb{F}_q^{(n-1)g} \times \mathbb{F}_2 \\
[D] &\longmapsto (\gamma_0, \ldots, \gamma_{d_1}, \beta_0, \ldots, \beta_{d_2 - 1}, \delta) \\
[0] &\longmapsto (0, \ldots, 0).
\end{aligned}$$

*Then $\mathcal{R}$ is an optimal representation for the elements of $T_n$, with the property that every element of $\mathrm{Im}\,\mathcal{R}$ has at most $n^g$ inverse images.*

*Proof.* It follows from Theorem 3.2 (iii) that

$$\deg h_{D,1} \leq \left\lfloor \frac{rn}{2} \right\rfloor \leq d_1 \text{ and } \deg h_{D,2} \leq \left\lfloor \frac{nr - 2g - 1}{2} \right\rfloor \leq d_2,$$

hence the polynomials can be written as claimed. Moreover, if $g$ is even and $r < g$, then

$$\deg h_{D,1} \leq \left\lfloor \frac{n(g-1)}{2} \right\rfloor \leq d_1 - 1 \text{ and } \delta = 0.$$

If $g = r$ is even, then $h_{D,1}$ is monic of degree $d_1$ and $\delta = 1$. If instead $g$ is odd and $r < g$, then

$$\deg h_{D,2} \leq \left\lfloor \frac{n(g-1) - 2g - 1}{2} \right\rfloor \leq d_2 - 1 \text{ and } \delta = 0.$$

Finally, if $g = r$ is odd, then $h_{D,2}$ is monic of degree $d_2$ and $\delta = 1$. Since $d_1 + d_2 + 1 = (n-1)g$, then $\mathrm{Im}\,\mathcal{R} \subseteq \mathbb{F}_q^{(n-1)g} \times \mathbb{F}_2$ in all cases. $\mathcal{R}$ is optimal since $(n-1)g\lceil \log_2 q \rceil + 1 = \lceil \log_2 |T_n| \rceil + O(1)$. Finally, the representation identifies at most $n^g$ elements by Theorem 3.2 (iv). $\square$

**Remark 3.6.** If one chooses to work only with divisors of the form $D = P_1 + \ldots + P_g - g\mathcal{O}$, then the last bit in the representation of Corollary 3.5 may be dropped and we have a representation of size $(n-1)g\lceil \log_2 q \rceil$. 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 e.g. for elliptic curves, where $r = 1$ if $D \neq 0$. Moreover, Lange [Lan04, Theorem 2.2] proved that for $g = 2$ and $n = 3$, all nontrivial elements of $T_3$ are represented by reduced divisors with $r = 2 = g$.

In the next theorem we establish some facts that we use for our decompression algorithm.

**Theorem 3.7.** *Let $[D] = [u, v] \in T_n$ with $D \in \mathrm{Div}^0_C$ a reduced divisor, and let $h_D = h_{D,1}(x) + y h_{D,2}(x) \in \mathbb{F}_q[x, y]$ be such that $\mathrm{div}(h_D) = \mathrm{Tr}(D)$. Write $D = D_1 + \ldots + D_t$, where $D_i \in \mathrm{Div}^0_C$ are reduced prime divisors defined over $\mathbb{F}_{q^n}$ with Mumford representation $[D_i] = [u_i, v_i]$. Then:*

- *(i) $h_{D,2} \equiv 0 \bmod u_i$ if and only if $w(D_i) = \varphi^j(D_k)$ for some $j \in \{0, \ldots, n-1\}$ and some $k \in \{1, \ldots, t\}$.*
- *(ii) Let $n \neq 2$. Then $w(D_i) = \varphi^j(D_i)$ for some $j \neq 0$ if and only if $D_i \in \mathrm{Pic}^0_C[2](\mathbb{F}_q)$.*
- *(iii) Let $n \neq 2$, $\ell, m \geq 0$, and assume that $D_i \neq w(D_i)$. Then $\mathrm{Tr}(D) = m \mathrm{Tr}(D_i) + \ell \mathrm{Tr}(w(D_i)) + \mathrm{Tr}(G)$ for some $G \in \mathrm{Div}^0_C$, where $\mathrm{Tr}(D_i), \mathrm{Tr}(w(D_i)) \not\leq \mathrm{Tr}(G)$ and $G$ has poles only at $\mathcal{O}$, if and only if $N(u_i)^{\min\{\ell, m\}}$ exactly divides $h_D$.*

*Proof.* (i) 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 2.1 (i).

(ii) We only prove the nontrivial implication. If $w(D_i) = \varphi^j(D_i)$ for some $j \neq 0$, then $u_i \in \mathbb{F}_q[x]$ and $-\nu = \nu^{\varphi^j}$ for all coefficients $\nu$ of $v_i$. Hence $\nu^2 = (\nu^2)^{\varphi^j}$, so $\nu \in \mathbb{F}_{q^{2j}} \cap \mathbb{F}_{q^n} = \mathbb{F}_q$. Therefore also $v_i \in \mathbb{F}_q[x]$, hence $w(D_i) = \varphi^j(D_i) = D_i \in \mathrm{Pic}^0_C(\mathbb{F}_q)$.

(iii) Let $\mathrm{Tr}(D) = m \mathrm{Tr}(D_i) + \ell \mathrm{Tr}(w(D_i)) + \mathrm{Tr}(G)$ for some divisor $G \in \mathrm{Div}^0_C$, with poles only at $\mathcal{O}$ and $\mathrm{Tr}(D_i)$, $\mathrm{Tr}(w(D_i)) \not\leq \mathrm{Tr}(G)$. Assume that $m \geq \ell$, since the proof of the other case is similar. Then

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

so $h_D = N(u_i)^\ell h_{D_i}^{m-\ell} h_G$ up to multiplication by a constant, hence $N(u_i)^\ell \mid h_D$. If $N(u_i)$ also divides $h_{D_i}^{m-\ell} h_G$, then $\mathrm{Tr}(D_i) + \mathrm{Tr}(w(D_i)) \leq (m - \ell) \mathrm{Tr}(D_i) + \mathrm{Tr}(G)$. Since $\mathrm{Tr}(w(D_i)) \not\leq \mathrm{Tr}(G)$ is prime by Lemma 2.1 (ii), then $\mathrm{Tr}(w(D_i)) = \mathrm{Tr}(D_i)$ and therefore $w(D_i) = \varphi^j(D_i)$ for some $j$. This yields a contradiction by (ii). Therefore, $N(u_i)^\ell$ exactly divides $h_D$.

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

$$\mathrm{Tr}(D) = m \mathrm{Tr}(D_i) + \ell \mathrm{Tr}(w(D_i)) + F$$

where $m = \ell + k$ and $\mathrm{Tr}(D_i), \mathrm{Tr}(w(D_i)) \not\leq \mathrm{div}(h) - k \mathrm{Tr}(D_i) =: F$. By Theorem 3.2 (iv), $F = \mathrm{Tr}(D) - m \mathrm{Tr}(D_i) - \ell \mathrm{Tr}(w(D_i)) = \mathrm{Tr}(G)$, where $G \in \mathrm{Div}^0_C$ is a reduced divisor with poles only at $\mathcal{O}$ of the form

$$G = D - \sum_{l=1}^{m} \varphi^{a_l}(D_i) - \sum_{l=1}^{\ell} \varphi^{b_l}(D_j)$$

for some $a_l, b_l \in \{0, \ldots, n-1\}$. $\qquad\square$

**Remark 3.8.** The results in this section 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 easy to check that we obtain a function $h_D$ with the same properties as in Theorem 3.2 and Corollary 3.5. Some caution is needed in adapting Theorem 3.7.

3.1. **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 $\mathrm{Cantor}(D_1, D_2) = (D_1 \oplus D_2, a)$. For completeness, we give Cantor's Algorithm in Algorithm 1. Lines 1–3 are the composition of the divisors to be added, and the result of this is reduced in lines 4–8.

---

**Algorithm 1** Cantor's Algorithm including rational function

---

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

---

The following iterative definition will allow us to compute $h_D$ with a Miller-style algorithm. For a function $h$ we denote by $h^\varphi$ the application of the Frobenius automorphism $\varphi : \overline{\mathbb{F}}_q \to \overline{\mathbb{F}}_q$ coefficientwise to the function $h$. The proof of the lemma is standard, and left to the reader.

**Lemma 3.9.** *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*

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

Algorithm 2 takes as an input the Mumford representation of $[D] \in T_n$ and the binary representation of $n - 1$, and returns the function $h_D$.

---

**Algorithm 2** 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** $j = s - 1, s - 2, \ldots, 1, 0$ **do**
 3: $\quad (R, a) \leftarrow \mathrm{Cantor}(R, \varphi^i(R)), h \leftarrow h \cdot h^{\varphi^i} \cdot a^{-1}, Q \leftarrow \varphi^i(Q), i \leftarrow 2i$
 4: $\quad$ **if** $n_j = 1$ **then**
 5: $\quad\quad (R, a) \leftarrow \mathrm{Cantor}(R, Q), h \leftarrow h \cdot u^{\varphi^i} \cdot a^{-1}, Q \leftarrow \varphi(Q), i \leftarrow i + 1$
 6: $\quad$ **end if**
 7: **end for**
 8: **return** $h$

---

**Theorem 3.10.** *Algorithm 2 computes $h_D$ correctly. It has an expected heuristic complexity of $O(n^{\log_2 3})$ operations in $\mathbb{F}_{q^n}$ asymptotically in $n$, or $O(g^4)$ operations in $\mathbb{F}_{q^n}$ asymptotically in $g$.*

*Proof.* Correctness of the algorithm follows from Lemma 3.9 for $[D] \in T_n$. We now analyze the complexity. The algorithm takes $s = \lfloor \log_2(n-1) \rfloor$ steps. We concentrate only on the "doublings" (line 3 of Algorithm 2), since they dominate the complexity of each step. For each $i$, the crucial operations are the execution of Cantor's Algorithm and the computation of $h^{(2i)}$ from $h^{(i)}$. According to [Can87], the algorithm has a complexity of $O(g^2 \log g)$ field operations in $\mathbb{F}_{q^n}$, which is negligible compared to the complexity of computing $h^{(2i)}$ from $h^{(i)}$, as we will see below. Notice that, while Cantor's original algorithm does not return the function $a$, it computes it implicitly while computing $h^{(2i)}$. Therefore, the complexity estimate of [Can87] applies also to the modified version of Cantor's Algorithm that we use.

Now we estimate the complexity of computing $h^{(2i)}$ from $h^{(i)}$. We start with a study of the shape of $a$, which we will use for the main complexity estimate. We assume that the input divisors to Cantor's Algorithm have $u$-polynomials and $v$-polynomials of degree $g$, which is true generically and which is also the most costly case algorithmically. Moreover, let us first assume that the $u$-polynomials of the input divisors are coprime. Then, at the end of the composition (after line 3 in Algorithm 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 1), and $u_i, v_i, a_i$ the values of $u, v, a$ after the $i$-th iteration of the while loop of Algorithm 1. Then, following through Algorithm 1, one can easily check that

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

and that both the numerator and the denominator of $a_i$ have $\lceil i/2 \rceil$ terms. Since the degree of $u$ decreases by at least 2 at each step (and, in fact, exactly by 2 in most cases, as observed already by Cantor [Can87]), and since we assume that $u_0$ has degree $2g$, we have to go through at most $\lceil g/2 \rceil$ reduction steps. Therefore, $a$ (output by the algorithm) has at most $\lceil \lceil g/2 \rceil/2 \rceil = \lceil g/4 \rceil$ terms in both the numerator and the denominator. Since we assume that $\deg v_i = 2g$ for all $i$, and computing modulo the curve equation (i.e. repeatedly replacing $y^2$ by $f(x)$), we get

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

where $b, d \in \mathbb{F}_{q^n}[x]$ are polynomials of degree at most $2g\lceil g/4 \rceil$ and $c, e \in \mathbb{F}_{q^n}[x]$ are polynomials of degree at most $2g(\lceil g/4 \rceil - 1)$, with equality expected heuristically. In particular, the degree of $b, c, d, e$ is in the order of $g^2$ and independent of $n$.

In the case that the $u$-polynomials of the input divisors to Cantor's Algorithm are not coprime, let $d = \deg a_0$. Then $\deg u_0 = 2g - 2d$, and hence Algorithm 1 only has to go through $\lceil g/2 \rceil - d$ reduction steps. In comparison with the case where the $u$-polynomials are coprime (in this case $a_0$ is a constant and $u_0$ has degree $2g$), here we start with an $a_0$ of degree $d$, but we have to go through $d$ less reduction steps. Therefore, with an analogous analysis to the above, we obtain that $a$ has shape (3) with $b, c, d, e \in \mathbb{F}_{q^n}[x]$ of degree about $g^2/2$.

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 that $h^{(i)}$ is of the form $h^{(i)} = h_1^{(i)} + yh_2^{(i)}$ with $h_1^{(i)}, h_2^{(i)} \in \mathbb{F}_{q^n}[x]$. By induction, it is easy to show that $\deg h_1^{(i)} \le \lfloor (i+1)g/2 \rfloor$ and $\deg h_2^{(i)} \le \lfloor ((i-1)g-1)/2 \rfloor$. Writing $h^{(i)} \cdot (h^{(i)})^{\varphi^i} \cdot (d+ye) = h_1' + yh_2'$, we obtain that $\deg h_1' \le 2\lfloor (i+1)g/2 \rfloor + 2g\lceil g/4 \rceil + 1$ and $\deg h_2' \le ig + 2g\lceil g/4 \rceil - 1$. We have

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

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, and we have $\deg(bh_1' - fch_2') \le 2\lfloor (i+1)g/2 \rfloor + 4g\lceil g/4 \rceil + 1$, $\deg(h_2' - ch_1^{(2i)}) \le ig + 2g\lceil g/4 \rceil - 1$, $\deg(b^2 - fc^2) \le 4g\lceil g/4 \rceil$, and $\deg b \le 2g\lceil g/4 \rceil$.

The most expensive multiplications involved are those by $h_1'$ and $h_2'$, since those have the largest degree, namely, it is in the order of $g^2 + ig$. Using Karatsuba multiplication, we can compute the numerators and denominators above in $O((g^2 + ig)^{\log_2 3})$ operations in $\mathbb{F}_{q^n}$. 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 computation of $h^{(2i)}$ takes $O(g^4 + g^3 i + (gi)^{\log_2 3})$ operations in $\mathbb{F}_{q^n}$.

Finally, we sum over the $s = \lfloor \log_2(n-1) \rfloor$ steps to obtain a total complexity of

$$O\left( \sum_{i=0}^{s} (g^4 + g^3 i + (gi)^{\log_2 3}) \right) = O\left( g^4 \log n + g^3 n + (gn)^{\log_2 3} \right)$$

from which the thesis follows. $\qquad\square$

**Remark 3.11.** It is also possible to determine the coefficients of $h_D$ by solving a linear system of size about $gn \times gn$. 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 2. Therefore, this method is preferable when $g \in \Omega(n^3/\log n)$.

3.2. **Compression and decompression algorithms.** We propose the compression and decompression algorithms detailed in Algorithms 3 and 4. We denote by lc the leading coefficient of a polynomial. We only discuss the case $n \ge 3$, since in the case $n = 2$ the representation consists of $u(x)$ as seen in Proposition 3.1.

The compression algorithm follows immediately from Corollary 3.5 and Algorithm 2. The strategy of the decompression algorithm is as follows. From the input $\alpha = \mathcal{R}(D)$, 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, for each $i \in \{1, \ldots, t\}$ we recover one of the Frobenius conjugates of $D_i$, which we denote by $D_i'$. The divisor $D_1' + \ldots + D_t'$ corresponds to the class $\mathcal{R}^{-1}(\alpha)$ by Theorem 3.2 (iv). We always compute a reduced representative $D_1' + \ldots + D_t'$ of the class $\mathcal{R}^{-1}(\alpha)$, as discussed in Remark 3.4.

**Theorem 3.12.**   (i) *The unique optimal representation from Corollary 3.5 can be computed by Algorithm 3 in $O(n^{\log_2 3})$ operations in $\mathbb{F}_{q^n}$ asymptotically in $n$, or $O(g^4)$ operations in $\mathbb{F}_{q^n}$ asymptotically in $g$.*

   (ii) *Decompression Algorithm 4 operates correctly, i.e. for any input $\mathcal{R}(D)$, where $[D] \in T_n$, it returns a reduced divisor $D'$ such that $[D'] \in T_n$ and $\mathcal{R}(D) = \mathcal{R}(D')$.*

   (iii) *The complexity of decompression Algorithm 4 is as follows. Asymptotically in $g$, it is expected to terminate after $O(g^{1+\log_2 3} \log g)$ operations in $\mathbb{F}_{q^n}$ for most inputs, and after at most $O(g^{2+\log_2 3})$ operations in $\mathbb{F}_{q^n}$ for any input. Asymptotically in $n$, it is expected to terminate after at most $O(n^{2+\log_2 3} \log n)$ operations in $\mathbb{F}_{q^n}$ for most inputs, and after at most $O(n^{3+\log_2 3})$ operations in $\mathbb{F}_{q^n}$ for any input. It has expected complexity $O(\log q)$ asymptotically in $q$ for most inputs, and at most $O(\log^2 q)$ for any input.*

---

**Algorithm 3** 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 2 and Remark 3.11)
 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: $h_{D,1} = \gamma_{d_1} x^{d_1} + \gamma_{d_1 - 1} x^{d_1 - 1} + \ldots + \gamma_1 x + \gamma_0$ 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: $h_{D,2} = \beta_{d_2} x^{d_2} + \beta_{d_2 - 1} x^{d_2 - 1} + \ldots + \beta_1 x + \beta_0$ 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**

---

*Proof.* (*i*) Correctness and optimality of the representation computed by Algorithm 3 follow from Corollary 3.5. The complexity of computing the representation is the same as the complexity of computing $h_D$, which is given in Theorem 3.10 and Remark 3.11.

(*ii*) Let $D = D_1 + \ldots + D_t$, where $D_i$ are reduced prime divisors defined over $\mathbb{F}_{q^n}$. If $D_i = \varphi^j(D_k)$ for some $k \neq i$, then $\mathcal{R}(D) = \mathcal{R}(\tilde{D})$ where $\tilde{D} = \sum_{j \neq i, k} D_j + 2D_i$. $\tilde{D}$ is reduced if $D_i \neq w(D_i)$. If that is the case, we may assume without loss of generality that

(4) $$D_i \neq \varphi^j(D_k) \text{ for any } k \neq i.$$

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

$$H_D(x) = \prod_{i=1}^{t} u_i^{1 + \varphi + \ldots + \varphi^{n-1}} = \prod_{i=1}^{m} U_i(x)^{e_i},$$

where $U_i \in \mathbb{F}_q[x]$ are irreducible and $U_i \neq U_j$ if $i \neq j$, $m \leq t$. Up to reindexing, $U_i = u_i$ if $u_i \in \mathbb{F}_q[x]$ and $U_i = N(u_i)$ otherwise, for $i \leq m$. 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 $\mathrm{Tr}(D_i) = nD_i$. Notice that by Lemma 2.1 (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 $\mathbb{F}_{q^n}[x]$. If $U_i$ is reducible in $\mathbb{F}_{q^n}[x]$, then $u_i \in \mathbb{F}_{q^n}[x]$ 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 $u_i(x)$. If $U \nmid h_{D,2}$ 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 $k(x)(h_{D,1}(x) + y h_{D,2}(x)) \equiv y + k(x) h_{D,1} \bmod U$. Since $h_{D,1} + y h_{D,2} \equiv 0 \bmod (U, y - V)$, then $V + k(x) h_{D,1} \equiv 0 \bmod U$, hence

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

---

**Algorithm 4** 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 $D \in \mathrm{Div}_C^0(\mathbb{F}_{q^n})$ such that $[D] \in T_n$ 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 $U_i \in \mathbb{F}_q[x]$ irreducible and pairwise 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**              $\triangleright$ $U_i$ comes from an $\mathbb{F}_q$-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**                                         $\triangleright$ $h_{D,2}(x) \equiv 0 \bmod U(x)$
22:          **if** $f(x) \equiv 0 \bmod U(x)$ **then**            $\triangleright$ $V(x) = 0$ 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**                           $\triangleright$ $V(x) \neq 0$ and $D_i \neq w(D_i)$
25:              compute $s, h_\Delta$ such that $h_D = U_i(x)^s h_\Delta$ and $U_i(x) \nmid h_\Delta$
26:              **if** $s < e_i/2$ **then**
27:                  $V(x) \leftarrow -h_{\Delta,1}(x) h_{\Delta,2}(x)^{-1} \bmod U(x)$
28:                  append $[U(x), V(x)]$ to $L, e_i - s$ times
29:                  append $[U(x)^\varphi, -V(x)^\varphi]$ to $L, s$ times
30:              **else**                            $\triangleright$ $s = e_i/2$
31:                  $V(x) \leftarrow \sqrt{f(x)} \bmod U(x)$
32:                  append $[U(x), V(x)], [U(x)^\varphi, -V(x)^\varphi]$ to $L, s$ times
33:              **end if**
34:          **end if**
35:      **end if**
36: **end for**                                                       $\triangleright$ Notation: $L = [D_1, \ldots, D_t]$
37: **return** $D = D_1 + \ldots + D_t$

---

Since $U \nmid h_{D,2}$, by Theorem 3.7 (i) no Frobenius conjugate of $w(D_i)$ appears among $D_1, \ldots, D_t$. Notice that in particular $D_i \neq w(D_i)$, hence $V \neq 0$. Therefore, $D_i$ appears in $D$ with multiplicity $e_i$ under assumption (4).

If $U \mid h_{D,2}$, it follows from Theorem 3.7 (i) that $w(D_i) = \varphi^j(D_k)$ for some $0 \leq j \leq n-1$ and $1 \leq k \leq t$. We distinguish the cases when $D_i = w(D_i)$ or $D_i \neq w(D_i)$. The case when

$D_i = w(D_i)$ is treated in lines 22–23 of the algorithm. Since $y^2 - f \in (U, y - V)$, then $V^2 \equiv f \bmod U$. Therefore $f \equiv 0 \bmod U$ if and only if $V = 0$, which is equivalent to $D_i = w(D_i)$ is equivalent to $v_i = 0$. Practically, one can decide whether $D_i = w(D_i)$ by checking whether $U \mid f$. If this is the case, it suffices to set $V = 0$. Since $U^{e_i}$ exactly divides $H_D$, $D_i$ and its Frobenius conjugates appear in $D$ with total multiplicity $e_i$. The divisor $D$ is reduced, therefore it must contain in its support $e_i$ distinct Frobenius conjugates of $D_i$, e.g. $D_i, \varphi(D_i), \dots, \varphi^{e_i-1}(D_i)$, each with multiplicity one.

The last case is treated in lines 25–33 of the algorithm. In this case $D_i \neq w(D_i)$, but $w(D_i) = \varphi^j(D_k)$ for some $k \in \{1, \dots, t\}$. This is equivalent to $U \mid h_{D,2}$ and $U \nmid f$, as we proved above. Since $n \neq 2$, then $k \neq i$ by Theorem 3.7 (ii). In addition, since $D$ is reduced, then $D_k \neq w(D_i)$, hence $D_i, D_k \notin \mathrm{Pic}_C^0(\mathbb{F}_q)$ and $U_i = N(U)$, $U \in \mathbb{F}_{q^n}[x] \setminus \mathbb{F}_q[x]$. Write $\mathrm{Tr}(D) = m\,\mathrm{Tr}(D_i) + \ell\,\mathrm{Tr}(w(D_i)) + \mathrm{Tr}(G)$ for some $m, \ell > 0$ such that $\mathrm{Tr}(D_i), \mathrm{Tr}(w(D_i)) \not\leq \mathrm{Tr}(G)$. By Theorem 3.7 (iii), $s := \min\{m, \ell\}$ may be computed as the exponent for which $U_i^s \mid h_D$ and $U_i^{s+1} \nmid h_D$. Equivalently, among $D_1, \dots, D_t$ there are at least $s$ Frobenius conjugates of $D_i$ (including $D_i$) and at least $s$ Frobenius conjugates of $w(D_i)$ (including $D_k$). No divisor can be a Frobenius conjugate of both, and for one among $D_i$ and $w(D_i)$ the multiset $\mathcal{D} = \{D_1, \dots, D_t\}$ contains exactly $s$ of its Frobenius conjugates. Remove $s$ of the Frobenius conjugates of $D_i$ and $s$ of the Frobenius conjugates of $w(D_i)$ from $\mathcal{D}$, and let $\Delta$ be the sum of the remaining divisors, counted with the multiplicity in which they appear in the multiset. Then $h_D = U_i^s h_\Delta$, where $h_\Delta = h_{\Delta,1} + y h_{\Delta,2}$ corresponds to the divisor $\Delta$. By Theorem 3.7 (i), $U \nmid h_{\Delta,2}$, since

$$\mathrm{Tr}(D_i) + \mathrm{Tr}(w(D_i)) \not\leq \mathrm{div}(h_\Delta) = \mathrm{Tr}(\Delta) = (m-s)\,\mathrm{Tr}(D_i) + (\ell-s)\,\mathrm{Tr}(w(D_i)) + \mathrm{Tr}(G).$$

If $s = e_i/2$, then the support of $D$ contains $e_i/2$ Frobenius conjugates of $D_i$ and $e_i/2$ Frobenius conjugates of $D_k$. Since it contains $e_i$ Frobenius conjugates of $D_i$ and $D_k$ in total, then $s = m = \ell$ and $V$ may be computed as $\sqrt{f} \bmod U$. Then $D$ contains exactly $e_i/2$ Frobenius conjugates of $[U, V]$ and $e_i/2$ Frobenius conjugates of $[U, -V]$. Notice that in this situation we do not need to distinguish between (Frobenius conjugates of) $D_i$ and $w(D_i)$, since they appear in $D$ with the same multiplicity. If $s < e_i/2$, then $h_D = U_i^\ell h_\Delta$ and $\Delta$ contains $e_i - 2s$ Frobenius conjugates of one among $D_i$ and $w(D_i)$. We already showed that $U \nmid h_{\Delta,2}$, hence the $V$ polynomial of the divisor which appears in $\Delta$ can be computed as $V = -h_{\Delta,1} h_{\Delta,2}^{-1} \bmod U$. In this case, $D$ contains $s$ Frobenius conjugates of $[U, -V]$ and $e_i - s$ Frobenius conjugates of $[U, V]$.

Finally, we show that the divisor returned by Algorithm 4 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 the case when $U \mid f$ we have $D_i = w(D_i)$. Since $D$ is reduced, then $e_i \leq n$, and if $e_1 \neq 1$ then $D_i \notin \mathrm{Pic}_C^0(\mathbb{F}_q)$. In particular, $D_i, \varphi(D_i), \dots, \varphi^{e_i-1}(D_i)$ are distinct. If $U \nmid f$, then we showed that $D_i, \varphi(D_i) \neq w(D_i)$ and $D_i \neq \varphi(D_i)$. The divisors $D_i = [U, V]$ and $w(\varphi(D_i)) = [U^\varphi, -V^\varphi]$ can be added with multiplicity greater than one since they are not 2-torsion and not one the involution of the other.

(iii) We assume that the degrees of $h_{D,1}$ and $h_{D,2}$ are maximal, which is the generic and also the most costly case. The complexity of the algorithm is dominated by the operations with polynomials. In the following, we analyze the complexity of each line of the algorithm that contains such an operation. We count operations in $\mathbb{F}_{q^n}$, unless stated otherwise.

Line 10: The most expensive operation is polynomial multiplication: We square $h_{D,1}$ and $h_{D,2}$, and multiply $h_{D,2}^2$ by $f$. Since $\deg h_{D,1} \leq \lfloor \frac{ng}{2} \rfloor$, $\deg h_{D,2} \leq \lfloor \frac{(n-2)g-1}{2} \rfloor$, and $\deg f = 2g+1$, Karatsuba multiplication takes $O((ng)^{\log_2 3})$ multiplications in $\mathbb{F}_q$, or $O(g^{\log_2 3})$ operations in $\mathbb{F}_{q^n}$.

Line 11: The factorization of $H_D$ over $\mathbb{F}_q$ takes an expected $O((ng)^{1+\log_2 3} \log(qng))$ operations in $\mathbb{F}_q$ by [GvzG99, Theorem 14.14], since $\deg H_D = ng$. This corresponds to $O(ng^{1+\log_2 3} \log(qng))$ operations in $\mathbb{F}_{q^n}$.

Next we study the loop over $i$ (lines 13–36). Denote the degree of $U_i$ by $k_i$ and the degree of the corresponding $U$, as computed in line 17, by $\kappa_i$. Notice that $k_i \leq ng$ and $\kappa_i \leq g$.

Lines 14 and 17: Factoring $U_i$ over $\mathbb{F}_{q^n}$ has an expected complexity of $O(k_i^{1+\log_2 3} \log(q^n k_i))$ operations, again by [GvzG99, Theorem 14.14].

Lines 18 and 19: Since the degree of $h_{D,2}$ is at most $\lfloor \frac{(n-2)g-1}{2} \rfloor$, reducing $h_{D,2} \bmod U$ has a complexity of $O((ng)^{\log_2 3})$ operations, using the division with remainder algorithm from [GvzG99, Theorem 9.6]. The same holds for $h_{D,1}$. To compute $h_{D,2}^{-1}$ modulo $U$, we perform a modular inversion of $h_{D,2} \bmod U$ in the ring $\mathbb{F}_{q^n}[x]/U(x)$. This takes $O(\kappa_i^{\log_2 3} \log \kappa_i)$ operations, as shown in [GvzG99, Corollary 11.8]. Finally we multiply $h_{D,1}$ and $h_{D,2}^{-1}$ and reduce the result modulo $U$, with a complexity of $O(\kappa_i^{\log_2 3})$.

Line 22: Computing $f \bmod U$, where $f$ has degree $2g+1$ and $U$ has degree $\kappa_i \leq g$, takes $O(g^{\log_2 3})$ operations, again according to [GvzG99, Theorem 9.6].

Line 25: In order to compute $s$ and $h_\Delta$, we have to perform a number of divisions with remainder, namely, we have to divide both $h_{D,1}$ and $h_{D,2}$ by $U_i$, $s+1$ times. Denoting by $s_i$ the $s$ of the $i$-th iteration of the loop, this takes $O(s_i(ng)^{\log_2 3})$ operations, see again [GvzG99, Theorem 9.6].

Line 27: Since the degree of $h_{\Delta,2}$ is not larger than that of $h_{D,2}$, this operation is not more expensive than that of line 19.

Line 31: The value of $f \bmod U$ was already computed in line 22. Then computing $\sqrt{f} \bmod U$ amounts to computing a square root in the finite field $\mathbb{F}_{q^n}[x]/U(x) = \mathbb{F}_{q^{n\kappa_i}}$. This can be done with the algorithm of Tonelli–Shanks [Ton91, Sha72], whose complexity depends on the shape of the field $\mathbb{F}_{q^{n\kappa_i}}$. In the worst case, where $q^{n\kappa_i} - 1$ is a power of two, the algorithm has expected complexity $O((n\kappa_i)^{2+\log_2 3}(\log q)^2)$ operations in $\mathbb{F}_{q^n}$.

Summarizing, the expected complexity asymptotically in $q$ is $O(\log q)$ if $\mathcal{T} = \mathcal{S} = \emptyset$, i.e. for most inputs, and $O(\log^2 q)$ in the worst case, i.e. for the most expensive inputs. Now we consider the complexity asymptotically in $n$ and $g$. The complexity of line 14 and 17 dominates that of line 19. The complexity of lines 18 dominates that of line 22, and it is the same as that of line 25 for those $i$ where $s_i = 0$ (or $s_i$ is bounded by any constant, in fact). Notice moreover that it is very unlikely that lines 25 and 31 are executed. For a fixed input, denote by $\mathcal{T} \subseteq \{1, \ldots, m\}$ the set of indices $i$ where the algorithm executes line 25 and $s_i > 0$, and by $\mathcal{S} \subseteq \{1, \ldots, m\}$ the set of indices $i$ where the algorithm executes line 31. Then the complexity of the loop in $n, g$ is

$$O\left( m(ng)^{\log_2 3} + \sum_{i=1}^m nk_i^{1+\log_2 3} \log(k_i) + \sum_{i \in \mathcal{T}} s_i(ng)^{\log_2 3} + \sum_{i \in \mathcal{S}} (n\kappa_i)^{2+\log_2 3} \right).$$

Now let us first assume that $\mathcal{T} = \mathcal{S} = \emptyset$. This is the case for most inputs. Then the complexity in $g$ of the loop is

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

This is largest in $g$ in the extreme case where $m = 1, k_1 = ng, \kappa_1 = g$. Then the complexity of the loop, asymptotically in $g$, is $O(g^{1+\log_2 3} \log g)$, which is also the complexity in $g$ of line 11. This is hence the overall asymptotic complexity in $g$ of the algorithm, assuming that $\mathcal{T} = \mathcal{S} = \emptyset$.

Asymptotically in $n$, the complexity of the loop is

$$O\left(mn^{\log_2 3} + \sum_{i=1}^{m} nk_i^{1+\log_2 3}\log(k_i)\right).$$

This complexity is between $O(n^2)$, in the case $m = ng, k_i = \kappa_i = 1$ for all $i$, and $O(n^{2+\log_2 3}\log n)$, in the case $m = 1, k_1 = ng, \kappa_1 = g$. Since both of these values are larger than the complexity in $n$ of lines 10 and 11, this is also the overall asymptotic complexity in $n$ of the algorithm, assuming that $\mathcal{T} = \mathcal{S} = \emptyset$.

Finally, we consider the worst case complexity. We will see that it is reached for inputs that require a square root computation (line 31) in every execution of the loop.

First we consider the case where $\mathcal{T}$ is maximal, i.e. $|\mathcal{T}| = m = ng$, and where $s_i$ is maximal in every iteration of the loop. This is the case when $\kappa_i = 1$, $h_{D,2} = U_i^{s_i}$, and $U_i^{s_i} \mid h_{D,1}$, hence $s_i = \lfloor \frac{(n-2)g-1}{2} \rfloor$, for all $i$. Then the execution of line 25 in every iteration of the loop has a total complexity of $O((ng)^{2+\log_2 3})$.

Now we assume that $\mathcal{S}$ is maximal, i.e. $|\mathcal{S}| = m$. In this case, the square root computations dominate the complexity of the entire algorithm. Asymptotically in $g$, when $m = 1, k_1 = ng, \kappa_1 = g$, the overall complexity is $O(g^{2+\log_2 3})$. Asymptotically in $n$, the worst case is when $m = ng, k_i = \kappa_i = 1$ for all $i$, where the overall complexity is $O(n^{3+\log_2 3})$.                                  $\square$

3.3. **Group operation.** An important question in the context of point compression is how to perform the group operation. For some compression methods for (hyper)elliptic curves, formulas or algorithms for performing the group operation in compressed coordinates are available. For example, the Montgomery ladder (see [Mon87]) computes the $x$-coordinate of an elliptic curve point $kP$ from the $x$-coordinate of $P$. This method may be generalized to genus 2 hyperelliptic curves (see [Gau07]). There is also an algorithm to compute pairings using the $x$-coordinates of the input points only (see [GL09]).

In such a situation, the crucial question is whether it is more efficient to perform the operation in the compressed coordinates, or to decompress, perform the operation in the full coordinates, and compress again. Implementation practice shows that it is usually more efficient to use the second method (at least when side-channel attack resistance is not crucial). For example, all recent speed records for scalar multiplication on elliptic curves have been set using algorithms that need the full point, in other words with the second approach, see e.g. [BDL$^+$12, LS12, OLAR13, FHLS14]. 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. Moreover, Galbraith and Lin show in [GL09] that for computing pairings, the second approach is faster whenever the embedding degree is greater than 2.

On the basis of these results, we recommend using the second method also when computing with compressed elements of a trace zero subgroup: Decompress the element, perform the operation in $\mathrm{Pic}_C^0(\mathbb{F}_{q^n})$, 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 divisors than for general divisors in $\mathrm{Pic}_C^0(\mathbb{F}_{q^n})$, due to a speed-up using the Frobenius endomorphism, as pointed out by Frey [Fre99] and studied in detail by Lange [Lan01, Lan04] and subsequently by Avanzi and Cesena [AC07].

## 4. REPRESENTATION FOR ELLIPTIC CURVES

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. Finding a rational function with a given principal divisor can also be made more efficient. For all these

reasons, the results and methods from Section 3 can be simplified and made explicit for elliptic curves.

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 then the group of all points $P$ with trace *equal* to zero. We consider only $n \geq 3$, and refer to [GM15] for the case $n = 2$.

**Notation 4.1.** Write $P_i = \varphi^i(P)$ for $i = 0, \ldots, n-1$. Let $\ell_i(x,y) = 0, i = 1, \ldots, n-2$, be the equation of the line passing through the points $P_0 \oplus \ldots \oplus P_{i-1}$ and $P_i$. Let $v_i(x,y) = 0, i = 1, \ldots, n-3$, be the equation of the vertical line passing through the point $P_0 \oplus \ldots \oplus P_i$.

The following is obtained from Theorems 3.2 and 3.7 in the case that the curve is elliptic. The proof that $h_P$ has the form claimed is an easy calculation, which is left to the reader.

**Corollary 4.2.** *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 $\ell_j$ and $v_j$ are the lines defined in Notation 4.1. Then:*

- *(i) $\mathrm{div}(h_P) = P_0 + \ldots + P_{n-1} - n\mathcal{O}$.*
- *(ii) $h_P(x,y) = h_{P,1}(x) + yh_{P,2}(x)$ for some $h_{P,1}, h_{P,2} \in \mathbb{F}_q[x]$.*
- *(iii) $H_P = h_{P,1}^2 - fh_{P,2}^2$ has degree $n$, and its zeros are exactly the $x$-coordinates of $P_0, \ldots, P_{n-1}$.*
- *(iv) $\deg h_{P,1} \leq \frac{n-1}{2}$ and $\deg h_{P,2} = \frac{n-3}{2}$.*
- *(v) If $Q$ is such that $h_P = h_Q$, then $Q = \varphi^j(P)$ for some $j \in \{0, \ldots, n-1\}$.*
- *(vi) $h_{P,2}(X) \neq 0$ for all $x$-coordinates of $P_0, \ldots, P_{n-1}$.*

Since the exact degree of $h_{P,2}$ is known, $h_P$ can be normalized by making $h_{P,2}$ monic, as in Corollary 3.5. One obtains the following optimal representation for trace zero points on an elliptic curve.

**Corollary 4.3.** *Let $n \geq 3$ prime, 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*

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

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

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

One also can give simplified compression and decompression algorithms.

---

**Algorithm 5** 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 $h_P(x,y) = h_{P,1}(x) + yh_{P,2}(x) \leftarrow \frac{\ell_1 \cdot \ldots \cdot \ell_{n-2}}{v_1 \cdot \ldots \cdot v_{n-3}}(x,y)$ (see Algorithm 7) 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** $(\gamma_0, \ldots, \gamma_{d_1}, \beta_0, \ldots, \beta_{d_2-1})$

---

Finally, we discuss how to compute $h_P$ for different values of $n$. Explicit formulas can be computed in the special cases $n = 3, 5$. We do this in Appendix A. For general $n$, a straightforward computation of $h_P$ is possible, since Corollary 4.2 contains an explicit formula given in terms of lines. Such a computation can be made more efficient by employing the usual divide and

---

**Algorithm 6** 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** $P = (X, Y)$

---

conquer strategy. Computing $h_P$ via a Miller-style algorithm analogous to Algorithm 2 is also possible. The latter is advantageous for medium and large values of $n$, while for small values of $n$ a straightforward computation using a divide and conquer approach seems preferable (unless explicit formulas are available).

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 7** 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: $\quad h \leftarrow h \cdot \frac{\ell_{R,Q}}{v_R}, \ R \leftarrow R \oplus Q, \ Q \leftarrow \varphi(Q), \ i \leftarrow 3$
  5: **end if**
  6: **for** $j = s - 2, s - 3, \ldots, 1, 0$ **do**
  7: $\quad h \leftarrow h \cdot h^{\varphi^i} \cdot \frac{v_{R + \varphi^i(R)}}{\ell_{w(R), w(\varphi^i(R))}}, \ R \leftarrow R \oplus \varphi^i(R), \ Q \leftarrow \varphi^i(Q), \ i \leftarrow 2i$
  8: $\quad$ **if** $n_j = 1$ **then**
  9: $\quad\quad h \leftarrow h \cdot \frac{\ell_{R,Q}}{v_R}, \ R \leftarrow R + Q, \ Q \leftarrow \varphi(Q), \ i \leftarrow i + 1$
 10: $\quad$ **end if**
 11: **end for**
 12: **return** $h$

---

The following is an adaptation of Theorem 3.12 to elliptic curves.

**Corollary 4.4.**     (*i*) *The execution of Algorithm 7, and therefore also of compression Algorithm 5, requires an expected $O(n^{\log_2 3})$ operations in $\mathbb{F}_{q^n}$.*
   (*ii*) $\mathcal{R}^{-1}(\mathcal{R}(P))$ *is one of the Frobenius conjugates of $P$. The expected complexity of decompression Algorithm 6 is $O(n^{1+\log_2 3} \log^2 n)$ operations in $\mathbb{F}_{q^n}$.*

*Proof.* (*ii*) Theorem 3.12 (iii) would give a complexity between $O(n^{2+\log_2 3} \log n)$ and $O(n^{3+\log_2 3})$ operations in $\mathbb{F}_{q^n}$. However, the situation here is much simpler, since the complexity of the algorithm depends only on the factorization of $H_D$, which splits into linear factors over $\mathbb{F}_{q^n}$. Therefore, we may apply the root finding algorithm of [GvzG99, Algorithm 14.15], which has an expected complexity of $O(n^{1+\log_2 3} \log^2 n)$ operations in $\mathbb{F}_{q^n}$. $\qquad\square$

**Remark 4.5.** A more careful analysis of Algorithm 7 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 $n > 20$.

## 5. Timings and comparison with other representations

Important achievements of this new representation are that it applies to 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 3.2 (iv). 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 [GM15]. While [Nau99] only applies to extension degree 3, [Sil05, GM15] can be made practical for $n = 3, 5$. The approach of [GM15] 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 [Lan04]. 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 $y$-coordinate 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 is done in [Sil05, Lan04]. In addition, our compression and decompression algorithms do not require a costly precomputation, such as that of the Semaev polynomial in [GM15] or the elimination of variables from a polynomial system in [Lan04].

In terms of efficiency, our compression algorithm is slower than all the other ones for elliptic curves, but our decompression algorithm is faster in all cases. For $g = 1$, the time for compression and decompression together is comparable for $n = 3$, and smaller for $n = 5$, than that of [GM15]. That is to say, the faster decompression makes up for the slower compression. Although in this paper we concentrate on the case of odd characteristic, our method can be adapted to fields of even characteristic, just like all other methods from [GM15, Sil05, Lan04, Nau99].

We now compare the efficiency of our algorithms with those of [GM15, Sil05, Lan04, Nau99] in more detail. The comparison of our method with that of [GM15] is on the basis of a precise operation count, complexity analysis, and our own Magma implementations. Notice that our programs are straightforward implementations of the methods described here and in [GM15], and they are only meant as an indication. No particular effort has been put into optimizing them, and clearly a special purpose implementation (e.g. choosing $q$ of a special shape) would produce better and more meaningful results. All computations were done with Magma version 2.19.3 [BCP97], running on one core of an Intel Xeon Processor X7550 (2.00 GHz). 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, Lan04] 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.

TABLE 1. Number of operations in $\mathbb{F}_q$ for compression/decompression of one point when $g = 1, n = 3$

| | |
|---|---|
| Compression | 2S+6M+1I |
| Compression [GM15] | 1M |
| Full decompression | 5S+5M+1I, 1 square root, 2 cube roots |
| Full decompression [GM15] | 4S+3M+2I, 1 square root, 2 cube roots, and 1 square root in $\mathbb{F}_{q^3}$ |
| Decompression $x$ only | 5S+4M+1I, 1 square root, 2 cube roots |
| Decompression $x$ only [GM15] | 4S+3M+2I, 1 square root, 2 cube roots |

TABLE 2. Average time in milliseconds for compression/decompression of one point when $g = 1, n = 3$

| $q$ | $2^{20} - 3$ | $2^{40} - 87$ | $2^{60} - 93$ | $2^{79} - 67$ |
|---|---|---|---|---|
| Compression | 0.01 | 0.03 | 0.03 | 0.04 |
| Compression [GM15] | 0.01 | 0.02 | 0.03 | 0.04 |
| Full decompression | 0.18 | 0.71 | 0.89 | 1.52 |
| Full decompression [GM15] | 0.84 | 7.62 | 10.62 | 17.58 |
| Decompression $x$ only | 0.15 | 0.63 | 0.87 | 1.40 |
| Decompression $x$ only [GM15] | 0.15 | 0.68 | 0.87 | 1.44 |

**Comparison and Timings for $g = 1, n = 3$.** We compare our method with the most efficient method from [GM15] (there called "compression in $t_i$") in terms of operations in Table 1 and timings in Table 2. 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 [GM15] requires fewer operations, but we could not observe a significant difference in the timings, probably due to insufficient accuracy of 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 [GM15] is much slower (roughly by a factor 10), due to the necessary square root extraction. This shows one major efficiency advantage of the approach that we follow in this paper: 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 paper and the one from [GM15] behave similarly.

Naumann [Nau99] does not give explicit compression or decompression algorithms, but he derives an equation for the trace zero subgroup that might be used for such. The equation is in the Weil restriction coordinates $x_0, x_1, x_2$ of the $x$-coordinate of a trace zero point, and it has degree 4 in $x_0$ and degree 3 in $x_1, x_2$. Therefore, it allows a representation in the coordinates $(x_0, x_1)$ or $(x_0, x_2)$, where decompression could be done by factoring a cubic polynomial in the missing coordinate, and then recomputing the $y$-coordinate as a square root. This is clearly more expensive than the decompression algorithm in this paper, which does not require polynomial factorization or square root extraction.

In [Sil05], compression is free. 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 paper. See [GM15, Section 5] for a more detailed discussion of the decompression algorithm from [Sil05].

TABLE 3. Number of operations/complexity for compression/decompression of one point when $g = 1, n = 5$

| | |
|---|---|
| Compression | 3S+18M+3I in $\mathbb{F}_{q^5}$ |
| Compression [GM15] | 5S+13M in $\mathbb{F}_q$ |
| Full decompression | $O(\log q)$ operations in $\mathbb{F}_q$ |
| Full decompression [GM15] | $O(\log q)$ 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 [GM15] | $O(\log q)$ operations in $\mathbb{F}_q$ |

TABLE 4. Average time in milliseconds for compression/decompression of one point when $g = 1, n = 5$

| $q$ | $2^{10} - 3$ | $2^{20} - 5$ | $2^{30} - 173$ | $2^{40} - 195$ |
|---|---|---|---|---|
| Compression | 0.21 | 0.25 | 0.46 | 0.80 |
| Compression [GM15] | 0.04 | 0.04 | 0.05 | 0.10 |
| Full decompression | 0.82 | 9.39 | 4.26 | 10.13 |
| Full decompression [GM15] | 5.89 | 17.90 | 30.21 | 63.60 |
| Decompression $x$ only | 0.77 | 9.36 | 4.01 | 9.82 |
| Decompression $x$ only [GM15] | 5.53 | 16.48 | 21.42 | 45.08 |

**Comparison and Timings for $g = 1, n = 5$.** A similar comparison for extension degree 5 (see Tables 3 and 4) shows that the compression algorithm proposed in this paper is less efficient than that of [GM15], 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 paper we have to factor one polynomial of degree 5 over $\mathbb{F}_{q^5}$, where the algorithm of [GM15] 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 paper performs much better than that of [GM15], regardless of whether we include the recovery of the $y$-coordinate. Notice that we again compare with the best method from [GM15], there called "compression/decompression in the $s_i$ with polynomial factorization".

In comparison to [Sil05], our compression algorithm is clearly less efficient, but our decompression method is much more efficient. The decompression algorithm of Silverberg involves resultant computations and the factorization of a degree 27 polynomial. For more detail, see [GM15, Section 6].

**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 2 and 4, we give in Table 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 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.

In each case, we choose the fastest method of computing $h_P$ during compression. As discussed in Remark 4.5, this is an iterative approach for $n = 7$, a divide and conquer approach for $n = 11, 13, 19$, and Algorithm 7 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.

TABLE 5. Average time in milliseconds for compression/decompression of one point when $g = 1, n > 5, \log_2 |T_n| \approx 160$

| $n$ | 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 |

TABLE 6. 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 |

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.

**Comparison and Timings for $g = 2, n = 3$.** We present timings for trace zero subgroups of 20, 30, 40, 50, 60 bits in Table 6. 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 for comparison purposes, we did not put much effort into producing suitable curves for larger trace zero subgroups.

The representation of [Lan04] 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 substitutes 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 clearly less efficient than our compression algorithm, since we only need to evaluate a number of expressions, 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 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 paper. 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 7 for trace zero subgroups of approximately 160 bits when $n = 5$ and $g = 5, 6, \ldots, 11$. The reason for this choice is simply that we are able to find suitable curves for these parameters. We stress again that the

TABLE 7. Average time in milliseconds for compression/decompression of one point when $n = 5, g \geq 5, \log_2 |T_n| \approx 160$

| $g$ | 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 |

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).

## 6. CONCLUSION

In this paper, we propose a representation of elements of the trace zero subgroup via rational functions. To the extent of our knowledge, this representation is the only one that applies to elliptic and hyperelliptic curves of any genus and field extensions of any prime degree. Our representation has convenient mathematical properties: It identifies well-defined 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 saving 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, if compression and decompression are considered together. 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, especially of pairing-based cryptographic systems.

## REFERENCES

[AC07]     R. M. Avanzi and E. Cesena, *Trace zero varieties over fields of characteristic 2 for cryptographic applications*, Proceedings of the First Symposium on Algebraic Geometry and Its Applications (SAGA '07), 2007, pp. 188–215.

[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.

[BCHL13] J. W. Bos, C. Costello, H. Hisil, and K. Lauter, *High-performance scalar multiplication using 8-dimensional GLV/GLS decomposition*, Cryptographic Hardware and Embedded Systems – CHES 2013 (G. Bertoni and J.-S. Coron, eds.), LNCS, vol. 8086, Springer, 2013, pp. 331–338.

[BCP97]   W. Bosma, J. Cannon, and C. Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), 235–265.

[BDL⁺12] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, *High-speed high-security signatures*, J. Cryptogr. Eng. **2** (2012), no. 2, 77–89.

[Bla02]   G. Blady, *Die Weil-Restriktion elliptischer Kurven in der Kryptographie*, Master's thesis, Universität GHS Essen, 2002.

[BV06]    P. S. L. M. Barreto and J. S. Voloch, *Efficient computation of roots in finite fields*, Des. Codes Crytogr. **39** (2006), no. 2, 275–280.

[Can87]   D. G. Cantor, *Computing in the Jacobian of a hyperelliptic curve*, Math. Comp. **48** (1987), no. 177, 95–101.

[Ces08]   E. Cesena, *Pairing with supersingular trace zero varieties revisited*, Available at http://eprint.iacr.org/2008/404, 2008.

[Ces10]   ———, *Trace zero varieties in pairing-based cryptography*, Ph.D. thesis, Università degli studi Roma Tre, Available at http://ricerca.mat.uniroma3.it/dottorato/Tesi/tesicesena.pdf, 2010.

[Die03]   C. Diem, *The GHS attack in odd characteristic*, Ramanujan Math. Soc. **18** (2003), no. 1, 1–32.

[Die11]   ———, *On the discrete logarithm problem in class groups of curves*, Math. Comp. **80** (2011), 443–475.

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

[EGO11]   P. N. J. Eagle, S. D. Galbraith, and J. Ong, *Point compression for Koblitz curves*, Adv. Math. Commun. **5** (2011), no. 1, 1–10.

[EGT11]   A. Enge, P. Gaudry, and E. Thomé, *An L(1/3) discrete logarithm algorithm for low degree curves*, J. Cryptology **24** (2011), 24–41.

[FHLS14]   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*, Topics in cryptology CT-RSA 2014, LNCS, vol. 8366, Springer, 2014, pp. 1–27.

[Fre99]   G. Frey, *Applications of arithmetical geometry to cryptographic constructions*, Proceedings of the 5th International Conference on Finite Fields and Applications, Springer, 1999, pp. 128–161.

[Gau07]   P. Gaudry, *Fast genus 2 arithmetic based on Theta functions*, J. Math. Cryptol. **1** (2007), 243–265.

[Gau09]   ———, *Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem*, J. Symbolic Comput. **44** (2009), no. 12, 1690–1702.

[GH99]   G. Gong and L. Harn, *Public-key cryptosystems based on cubic finite field extensions*, IEEE Trans. Inform. Theory **45** (1999), no. 7, 2601–2605.

[GHS02]   P. Gaudry, F. Hess, and N.P. Smart, *Constructive and destructive facets of Weil descent*, J. Cryptology **15** (2002), no. 1, 19–46.

[GL09]   S. D. Galbraith and X. Lin, *Computing pairings using x-coordinates only*, Des. Codes Crytogr. **50** (2009), no. 3, 305–324.

[GLS11]   S. D. Galbraith, X. Lin, and M. Scott, *Endomorphisms for faster elliptic curve cryptography on a large class of curves*, J. Cryptology **24** (2011), no. 3, 446–469.

[GLV01]   R. P. Gallant, R. J. Lambert, and S. A. Vanstone, *Faster point multiplication on elliptic curves with efficient endomorphisms*, Advances in Cryptology: Proceedings of CRYPTO '01 (J. Kilian, ed.), LNCS, vol. 2139, Springer, 2001, pp. 190–200.

[GM15]   E. Gorla and M. Massierer, *Point compression for the trace zero subgroup over a small degree extension field*, Des. Codes Crytogr. **75** (2015), no. 2, 335–357.

[GvzG99]   J. Gerhard and J. von zur Gathen, *Modern computer algebra*, Cambridge University Press, Cambridge, 1999.

[HSS01]   F. Hess, G. Seroussi, and N. P. Smart, *Two topics in hyperelliptic cryptography*, Proceedings of SAC '01 (S. Vaudenay and A. M. Youssef, eds.), LNCS, vol. 2259, Springer, 2001, pp. 181–189.

[Kob91]   N. Koblitz, *CM-curves with good cryptographic properties*, Advances in Cryptology: Proceedings of CRYPTO '91 (J. Feigenbaum, ed.), LNCS, vol. 576, Springer, 1991, pp. 179–287.

[Lan01]   T. Lange, *Efficient arithmetic on hyperelliptic curves*, Ph.D. thesis, Univerität GHS Essen, Available at http://www.hyperelliptic.org/tanja/preprints.html, 2001.

[Lan04]   ———, *Trace zero subvarieties of genus 2 curves for cryptosystem*, Ramanujan Math. Soc. **19** (2004), no. 1, 15–33.

[Lan05]   ———, *Formulae for arithmetic on genus 2 hyperelliptic curves*, Appl. Algebra Engrg. Comm. Comput. **15** (2005), 295–328.

[LS12]   P. Longa and F. Sica, *Four-dimensional Gallant–Lambert–Vanstone scalar multiplication*, Advances in Cryptology: Proceedings of ASIACRYPT '12 (X. Wang and K. Sako, eds.), LNCS, vol. 7658, Springer, 2012, pp. 718–739.

[LV00]   A. K. Lenstra and E. R. Verheul, *The XTR public key system*, Advances in Cryptology: Proceedings of CRYPTO '00 (M. Bellare, ed.), LNCS, vol. 1880, Springer, 2000, pp. 1–19.

[Mil04]   V. S. Miller, *The Weil pairing, and its efficient calculation*, J. Cryptology **17** (2004), no. 4, 235–261.

[Mon87]   P. L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Math. Comp. **48** (1987), no. 177, 243–264.

[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.

[OLAR13]   T. Oliveira, J. López, D. F. Aranha, and F. Rodríguez-Henríquez, *Lambda coordinates for binary elliptic curves*, Cryptographic Hardware and Embedded Systems – CHES 2013 (G. Bertoni and J.-S. Coron, eds.), LNCS, vol. 8086, Springer, 2013, pp. 311–330.

[RS02]    K. Rubin and A. Silverberg, *Supersingular abelian varieties in cryptology*, Advances in Cryptology: Proceedings of CRYPTO '02 (M. Yung, ed.), LNCS, vol. 2442, Springer, 2002, pp. 336–353.

[RS03]    _____, *Torus-based cryptography*, Advances in Cryptology: Proceedings of CRYPTO '03 (D. Boneh, ed.), LNCS, vol. 2729, Springer, 2003, pp. 349–365.

[RS09]    _____, *Using abelian varieties to improve pairing-based cryptography*, J. Cryptology **22** (2009), no. 3, 330–364.

[Sha72]   D. Shanks, *Five number-theoretic algorithms*, Proceedings of the Second Manitoba Conference on Numerical Mathematics, 1972, pp. 51–70.

[Sil05]   A. Silverberg, *Compression for trace zero subgroups of elliptic curves*, Trends Math. **8** (2005), 93–100.

[SS95]    P. Smith and C. Skinner, *A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms*, Advances in Cryptology: Proceedings of ASIACRYPT '94 (J. Pieprzyk and R. Safavi-Naini, eds.), LNCS, vol. 917, Springer, 1995, pp. 357–364.

[Sta04]   C. Stahlke, *Point compression on Jacobians of hyperelliptic curves over* $\mathbb{F}_q$, Available at `http://eprint.iacr.org/2004/030`, 2004.

[Ton91]   A. Tonelli, *Bemerkung über die Auflösung quadratischer Congruenzen*, Göttinger Nachrichten (1891), 344–346.

[Was08]   L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, second ed., Discrete Mathematics and its Applications, Chapman & Hall/CRC, Boca Raton–London–New York, 2008.

[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.

## Appendix A. Explicit equations

We compute explicit equations for compression and decompression for the cases when $g = 1$ and $n = 3, 5$, or $g = 2$ and $n = 3$. We give explicit formulas for compression, while for decompression we explicitly compute a low degree polynomial, whose roots give the result of the decompression.

In addition to making the computation more efficient, the results contained in this appendix allow us to perform precise operation counts, and thus to compare our method to the other existing compression methods in Section 5. When computing complexities, we count squarings (S), multiplications (M), and inversions (I) in $\mathbb{F}_q$, but not additions or multiplications by constants.

A.1. **Explicit equations for** $g = 1, n = 3$. In this case $h_P = \ell_1$ is a line through the points $P, \varphi(P), \varphi^2(P)$. 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

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

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 $\mu$ of small size, see [Lan04, Section 3.1]. When working with a field extension where $3 \nmid q - 1$, one may use a normal basis, which yields similar but denser equations.

**Compression.** If $P = (X, Y) \notin E(\mathbb{F}_q)$, then the equation of $h_P = \ell_1$ is

$$h_P = y + \gamma_1 x + \gamma_0$$

and $\mathcal{R}(P) = (\gamma_0, \gamma_1) \in \mathbb{F}_q^2$. Let

(5)
$$X = X_0 + X_1\zeta + X_2\zeta^2$$
$$Y = Y_0 + Y_1\zeta + Y_2\zeta^2$$

then a simple computation yields

$$\gamma_1 = \frac{c_1 X_1^2 Y_1 + c_2 X_2^2 Y_2}{c_1 X_1^3 + c_2 X_2^3}$$
$$\gamma_0 = -\gamma_1 X_0 - Y_0,$$

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. Hence compression takes 2S+6M+1I in $\mathbb{F}_q$.

When $P \in E(\mathbb{F}_q)$, the line $\ell_1$ is a tangent and we have

$$
\gamma_1 = \frac{3X^2 + A}{2Y}
$$
$$
\gamma_0 = -\gamma_1 X - Y.
$$

Notice that such points are in $E[3](\mathbb{F}_q)$ and therefore very few.

**Decompression.** This algorithm computes the polynomial $H_P$ and its roots over $\mathbb{F}_{q^3}$. We have

$$
H_P(x) = x^3 - \gamma_1^2 x^2 + (A - 2\gamma_0\gamma_1)x - \gamma_0^2 + B.
$$

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 (5), we get

$$
\begin{aligned}
(6) \qquad \gamma_1^2 &= X + X^q + X^{q^2} &&= 3X_0 \\
A - 2\gamma_0\gamma_1 &= X^{1+q} + X^{1+q^2} + X^{q+q^2} &&= 3X_0^2 - 3\mu X_1 X_2 \\
\gamma_0^2 - B &= X^{1+q+q^2} &&= X_0^3 - 3\mu X_0 X_1 X_2 + \mu X_1^3 + \mu^2 X_2^3.
\end{aligned}
$$

Hence one can solve system (6) over $\mathbb{F}_q$, to recover $(X_0, X_1, X_2)$. Since the solutions of the system are exactly the Frobenius conjugates of $X$, 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 [GM15, Section 5]). Notice that, since this system is so simple, this is more efficient than factoring $H_P$ 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 $\mathbb{F}_q$.

A.2. **Explicit equations for $g = 1, n = 5$.** 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 $\ell_1, \ell_2, \ell_3$, respectively. We have

$$
h_P = \frac{\ell_1 \ell_2 \ell_3}{v_1 v_2} = (\gamma_2 x^2 + \gamma_1 x + \gamma_0) + y(x + \beta_0),
$$

where

$$
\begin{aligned}
\gamma_2 &= -\lambda_1 - \lambda_2 - \lambda_3 \\
\beta_0 &= -\lambda_2 \gamma_2 + \lambda_1 \lambda_3 - X^{q^2} \\
\gamma_1 &= -\lambda_2 \beta_0 - \gamma_2 X^{q^2} + \lambda_1 X + \lambda_3 X^{q^3} - Y - Y^{q^2} - Y^{q^3} \\
\gamma_0 &= \gamma_1(\lambda_2^2 - X^{q^2}) + \gamma_2((X + X^q)(X + X^q - X^{q^2} - 2\lambda_1^2 + \lambda_2^2) + \lambda_1^4 + A + \lambda_1^2 X^{q^2}) \\
&\quad + \lambda_1 \lambda_2 \lambda_3 (X + X^{q^2} + X^{q^3}) - \lambda_1 \lambda_2 Y^{q^3} - \lambda_1 \lambda_3 Y^{q^2} - \lambda_2 \lambda_3 Y + \lambda_3 \lambda_1^2 \lambda_2^2 + \lambda_1^3 \lambda_2^2 + \lambda_1^2 \lambda_2^3.
\end{aligned}
$$

Computing $\lambda_1, \lambda_2, \lambda_3$ takes a total of 3M+3I in $\mathbb{F}_{q^5}$. Then, $\beta_0, \gamma_0, \gamma_1, \gamma_2$ can be computed with a total of 3S+15M in $\mathbb{F}_{q^5}$. 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 $H_P(x) = x^5 - S_1 x^4 + S_2 x^3 - S_3 x^2 + S_4 x - S_5$, which takes $O(\log_2 q)$ operations in $\mathbb{F}_q$. Finally, recovering $Y$ costs 1S+3M+1I in $\mathbb{F}_{q^5}$.

A.3. **Explicit equations for** $g = 2, n = 3$. We assume $2, 3 \nmid |\operatorname{Pic}_C^0(\mathbb{F}_{q^3})|$ and that the characteristic of $\mathbb{F}_q$ is not equal to 2 or 5. A simple transformation yields a curve equation of the shape

$$C : y^2 = x^5 + f_3 x^3 + f_2 x^2 + f_1 x + f_0.$$

We assume that $C$ is given in this form, which slightly simplifies the equations. Formulas for the general case can be worked out similarly.

The trace zero variety of hyperelliptic curves of genus 2, with respect to a degree 3 base field extension, was studied in detail by Lange [Lan01, Lan04]. One of her results is that the Mumford representation of all non-trivial elements of $T_3$ has a $u$-polynomial of degree 2.

**Theorem A.1** ([Lan04, Theorem 2.2]). *Assume that $C$ has genus 2 and that $2, 3 \nmid |\operatorname{Pic}_C^0(\mathbb{F}_{q^3})|$. Then all non-trivial elements of $T_3$ are represented by reduced divisors of the form*

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

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

**Corollary A.2.** *Assume that $C$ has genus 2 and that $2, 3 \nmid |\operatorname{Pic}_C^0(\mathbb{F}_{q^3})|$. Then all non-trivial elements of $T_3$ are represented by reduced divisors of the form $D = P_1 + P_2 - 2\mathcal{O} \notin \operatorname{Div}_C(\mathbb{F}_q)$, 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 $P_1 \neq P_2, \varphi(P_2), \varphi^2(P_2), w(\varphi(P_2)), w(\varphi^2(P_2))$,*
*(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.* It is easy to check that (i)-(iii) are mutually exclusive, and that one must be in one of these situations. We now show that $D + \varphi(D)$ is semi-reduced and $u \nmid h_{D,2}$. If we are in case (ii), then clearly $D + \varphi(D)$ is semi-reduced. By contradiction assume that $h_{D,2} \equiv 0 \bmod u$. Let $P_j = (X_j, Y_j), j = 1, 2$. $P_j - \mathcal{O} \in \operatorname{Div}_C(\mathbb{F}_{q^3})$ is a reduced prime divisor. Since $h_{D,2}(X_j) = 0$, by Theorem 3.7 (i) 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 $D = P_1 + P_2 - 2\mathcal{O} \in \operatorname{Div}_C(\mathbb{F}_q)$, which contradicts Theorem A.1.

Assume now that we are in case (iii). Since $D$ is prime, by Theorem 3.7 (i), $u \mid h_{D,2}$ if and only if $w(D) = \varphi^i(D)$ for some $i = 1, 2$. By contradiction, assume this is the case. Then either $w(P_1) = \varphi^i(P_1)$ or $w(P_1) = \varphi^{i+3}(P_1)$. Hence $X = X^{q^j} \in \mathbb{F}_{q^6} \cap \mathbb{F}_{q^j} \subseteq \mathbb{F}_{q^2}$ and $Y = -Y^{q^j} \in \mathbb{F}_{q^6} \cap \mathbb{F}_{q^{2j}} \subseteq \mathbb{F}_{q^2}$ for some $j \in \{i, i+3\}$. This shows that $D \in \operatorname{Div}_C(\mathbb{F}_{q^2}) \cap \operatorname{Div}_C(\mathbb{F}_{q^3}) = \operatorname{Div}_C(\mathbb{F}_q)$, which contradicts Theorem A.1. Therefore $u \nmid h_{D,2}$ and $D + \varphi(D) = P_1 + \varphi(P_1) + \varphi^3(P_1) + \varphi^4(P_1) - 4\mathcal{O}$ is semi-reduced. Notice that $P_1 \neq w(\varphi(P_2))$ and $P_2 \neq w(\varphi(P_1))$, since $D$ is reduced. $\square$

**Compression.** We consider elements $0 \neq [D] = [u,v] \in T_3$, $D = P_1 + P_2 - 2\mathcal{O}$ with $P_1 \neq w(\varphi(P_2))$, $w(\varphi^2(P_2))$ and $u, u^\varphi$ coprime. The special cases can be worked out separately, and we do not treat them here.

**Proposition A.3.** *Let $0 \neq [D] = [u,v] \in T_3$, $D = P_1 + P_2 - 2\mathcal{O}$ with $P_1 \neq w(\varphi(P_2))$, $w(\varphi^2(P_2))$ and $\gcd(u, u^\varphi) = 1$. 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 A.2. By Theorem 3.2 (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 Corollary A.2, after multiplication by a constant we have $h_D = y - \gamma(x)$ where $\gamma \in \mathbb{F}_q[x]$ of degree 3. If $P_i = (X_i, Y_i)$, 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 $\leq 3$ 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 it is the unique polynomial $V$ of degree $< \deg(uu^\varphi) = 4$ such that $V \equiv v \bmod u$ and $V \equiv v^\varphi \bmod u^\varphi$. Keeping in mind that $u, u^\varphi$ are coprime, and using the Chinese Remainder Theorem (or following the explicit formulas in [Lan05]), we get

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

as claimed. $\qquad\square$

Denoting $u(x) = x^2 + u_1 x + u_0$ and $v(x) = v_1 x + v_0$, 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

$$
\begin{aligned}
d &= (U_1 V_0 - U_0 V_1)^{-1} \\
\beta_0 &= ((u_0 u_1^q - u_0^q u_1)U_1 - U_0^2)d \\
\gamma_0 &= ((u_0 v_0^q - u_0^q v_0)U_0 + (u_0^q u_1 v_0 - u_0 u_1^q v_0^q - u_0^{q+1} V_1)U_1)d \\
\gamma_1 &= ((u_0 v_1^q - u_0^q v_1)U_0 + (u_1^q v_0 + u_0^q v_1^q)u_1 U_1 + (u_0^q u_1 - u_0 u_1^q)V_0 + (u_0 v_1 + u_1 v_0^q)(u_1^{2q} - u_1^{q+1}))d \\
\gamma_2 &= (((u_1 + u_1^q)U_1 - U_0)V_0 - (u_0 u_1 - u_0^q u_1^q)V_1)d.
\end{aligned}
$$

Computing these values in the 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.

**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 $H_D = x^6 - S_1 x^5 + S_2 x^4 - S_3 x^3 + S_4 x^2 - S_5 x + S_6$. In almost all cases we are decompressing a divisor of the shape that we consider above for compression. $H_D$ either splits over $\mathbb{F}_q$ into two factors of degree 3, or it is 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 all cases, we then compute the

corresponding $v$-polynomials. It follows that the overall complexity of decompression is $O(\log q)$ operations in $\mathbb{F}_q$.

Elisa Gorla, Institut de mathématiques, Université de Neuchâtel, Rue Emile-Argand 11, 2000 Neuchâtel, Switzerland
*E-mail address*: elisa.gorla@unine.ch

Maike Massierer, Mathematisches Institut, Universität Basel, Rheinsprung 21, 4051 Basel, Switzerland
*E-mail address*: maike.massierer@inria.fr