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. Such cryptographic systems work in finite groups which must satisfy three basic requirements: Computing with elements of the group must be efficient, the discrete logarithm problem (DLP) in the group must be hard, and there must be a convenient and compact representation for the group 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]. Although the trace zero subgroup is a proper subgroup of the $\mathbb{F}_{q^n}$-rational points of the Jacobian of the curve, it can be shown that the DLP in both groups has the same complexity. Therefore, from a mathematical point of view, trace zero variety cryptosystems may be regarded as the (hyper)elliptic curve analog of torus-based cryptosystems such as LUC [SS95], Gong-Harn [GH99], XTR [LV00], and CEILIDIH [RS03].

The trace zero subgroup is of particular interest in the context of pairing-based cryptography. Rubin and Silverberg have shown in [RS02, RS09] that the security of pairing-based cryptosystems can be improved by using abelian varieties of dimension greater than one in place of elliptic curves. Jacobians of hyperelliptic curves and trace zero varieties are therefore the canonical examples for such applications. E.g., over a field of characteristic 3 the known examples of groups

2010 *Mathematics Subject Classification.* primary: 14G50, 11G25, 14H52, secondary: 11T71, 14K15.

Key words and phrases. Elliptic and hyperelliptic curve cryptography, pairing-based cryptography, discrete logarithm problem, trace zero variety, efficient representation, point compression.

The authors were supported by the Swiss National Science Foundation under grant no. 123393.

with highest security parameter (i.e., 7.5) come from trace zero subgroups relative to a field extension $\mathbb{F}_{q^5}|\mathbb{F}_q$.

Scalar multiplication in the trace zero subgroup is particularly efficient, due to a speed-up using the Frobenius endomorphism, see [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, FHLS13, BCHL13]. In[AC07], Avanzi and Cesena show in that trace zero subgroups in general deliver better scalar multiplication performance than elliptic curves, and trace zero subgroups constructed from elliptic curves over degree 5 extension fields are almost 3 times faster than elliptic curves for the same group size. They conclude that trace zero subgroups are very interesting groups for the design of cryptographic systems based on the discrete logarithm problem due to their vastly superior performace, but that 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 approach 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 [GM14] for elliptic curves with respect to base field extensions of degree 3 and 5. They all build on the standard compact representation for elliptic and hyperelliptic curves, where a divisor class is represented by the u-polynomial of the Mumford representation, possibly together with some extra bits, see [HSS01, Sta04]. This is a generalization of the standard efficient representation for elliptic curves, which consists only of the x-coordinate of a point (notice that the y-coordinate can be recomputed from the curve equation up to sign). 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 cryptosystem, where the only operation required is scalar multiplication of points, this enables us to compute with equivalence classes of trace zero elements, and no extra bits are required to distinguish between the different representatives.

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

The paper is organized as follows: In Section 2 we give some preliminaries on (hyper)elliptic curves, the trace zero variety, and compact 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. Then we give explicit equations for the most studied cases $g = 1, n = 3, 5$ and $g = 2, n = 3$ in Section 5, and present some implementation results, as well as a detailed comparison with the other compression methods, in Section 6.

Acknowledgements. We thank Tanja Lange for bringing to our attention the work of Naumann, and we are grateful to the mathematics department of the University of Zürich for access to their computing facilities.

2. Preliminaries

We start by recalling the most important definitions 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. We assume that $\mathbb{F}_q$ does not have characteristic 2. Then C can be given by an 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 Div_C the group of divisors of C. Let w be the involution

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

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 φ extend to group homomorphisms on 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 Div_C.

Let $D_1 = a_1P_1 + \ldots + a_kP_k - a\mathcal{O}, D_2 = b_1P_1 + \ldots + b_kP_k - b\mathcal{O} \in \mathrm{Div}_C$, $a_i, b_i, a, b \in \mathbb{Z}_+ \cup \{0\}$, 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 Pic^0_C 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 Pic^0_C and $D_1 \sim D_2$ for $[D_1] = [D_2]$. An $\mathbb{F}_{q^n}$-rational divisor class $[D]$ is an equivalence class of $\mathbb{F}_{q^n}$-rational divisors. The subgroup of Pic^0_C consisting of the $\mathbb{F}_{q^n}$-rational divisor classes is denoted by $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ and has approximately q^{ng} elements for large q, by the Theorem of Hasse-Weil.

Often in cryptography, the Jacobian variety is used. The Jacobian of C is the abelian variety Jac_C that contains C and has the property that for every extension field $\mathbb{F}_{q^n}$ of $\mathbb{F}_q$, the groups $\mathrm{Jac}_C(\mathbb{F}_{q^n})$ and $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ are isomorphic. For our purposes, it will suffice to work with the Picard group.

It follows from the Riemann-Roch Theorem that every divisor class can be represented by a unique *reduced divisor* $D = P_1 + \ldots + P_r - r\mathcal{O}$ such that:

- $r \in \{0, \ldots, g\}$
- $P_i \in C \setminus \{\mathcal{O}\}$
- $P_i \neq w(P_j)$ for $i \neq j$ (while $P_i = P_j$ is possible).

Notice that $r = 0$ if and only if $[D] = 0$. We have $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ if and only if $\varphi^n(\{P_1, \ldots, P_r\}) = \{P_1, \ldots, P_r\}$. For any divisors D_1 and D_2, we denote by $D_1 \oplus D_2$ the reduced divisor such that $D_1 \oplus D_2 \sim D_1 + D_2$.

When C is an elliptic curve (i.e. $g = 1$), then each non-zero element of Pic^0_C is uniquely represented by a divisor of the form $P - \mathcal{O}$ with $P \in C$. In fact, we have $C \cong \mathrm{Pic}^0_C$ as groups via $P \mapsto [P - \mathcal{O}]$, and the addition $\oplus$ is the usual addition of elliptic curve points. For elliptic curves, we denote a divisor class by the unique corresponding $P \in C$. In particular, we denote $0 \in \mathrm{Pic}^0_C$ by the point $\mathcal{O}$ (notice that this is not the reduced representation of the divisor zero).

Another very useful representation of divisor classes is the *Mumford representation.* An element $[D] \in \mathrm{Pic}^0_C$ with $D = P_1 + \ldots + P_r - r\mathcal{O}$ a reduced divisor and $P_i = (X_i, Y_i)$ is represented by a pair of polynomials $[u(x), v(x)]$ where

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

There is, in fact, a one-to-one correspondence between reduced divisors D and pairs $[u, v]$ such that u is monic, $\deg v < \deg u \leq g$, and $u \mid v^2 - f$: Given the divisor D, then $u(x) = \prod_{i=1}^{r}(x - X_i)$ and $v(x)$ is the unique polynomial such that $v(X_i) = Y_i$ with multiplicity equal to the multiplicity of P_i in D. The polynomial $v(x)$ may be computed by solving a linear system. Conversely, given the Mumford representation $[u, v]$, the corresponding reduced divisor is the D with defining ideal $I_D = (u(x), y - v(x))$. Notice that the curve equation $y^2 - f$ becomes superfluous, since $y^2 - f \in (u, y - v)$. If C is an elliptic curve, then the Mumford representation of $P = (X, Y) \in C$ is $[x - X, Y]$. A convenient property of the Mumford representation is that $\mathbb{F}_{q^n}$-rationality of divisor classes becomes easily visible: We have $[u, v] \in \mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ if and only if $u, v \in \mathbb{F}_{q^n}[x]$. It follows from the definition that the Mumford representation of $[0]$ is $[1, 0]$.

Let $\mathbb{F}_{q^n}|\mathbb{F}_q$ be a field extension, $n \geq 1$. A divisor $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ with Mumford representation $[u, v]$ is *prime* if $u \in \mathbb{F}[x]$ is an irreducible polynomial. Notice that being prime depends on the choice of $\mathbb{F}_{q^n}$. Sometimes it will be convenient to write a divisor as a sum of prime divisors: $D = D_1 + \ldots + D_t$, $D_i \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ prime. Notice that it may be $D_i = D_j$ for $i \neq j$, and $D_1, \ldots, D_t$ are unique up to permutation.

The Mumford representation is particularly useful when computing with divisor classes, and all algorithms given in this paper make use of this representation. Cantor's Algorithm performs the addition of divisor classes in the Mumford representation. An easy modification, as given in Algorithm 1, computes not only $D_1 \oplus D_2$, but also a function a such that $D_1 + D_2 = D_1 \oplus D_2 + \mathrm{div}(a)$. For elliptic curves, addition formulas are easier to use and more efficient than Cantor's Algorithm. Also for genus 2 curves, there exist explicit addition formulas that are more efficient than Cantor's Algorithm, see [Lan05].

Remark 2.1. The Mumford representation can also be defined for *semi-reduced* divisors. These are divisors that satisfy all the same properties as reduced divisors, except the condition that $r \leq g$. In that case, the degrees of u and v may be greater than g.

For more details and mathematical background on elliptic and hyperelliptic curves, see [Was08, ACD$^+$06]. The literature also explains the more general theory including curves over characteristic 2 fields. By making the necessary adjustments, the content of this paper carries over to the characteristic 2 case, and we stick to characteristic different from 2 only for convenience and ease of exposition. See also Remark 3.4.

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

The following will be useful in the sequel.

Lemma 2.2. *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*) $D \in \mathrm{Div}_C(\mathbb{F}_{q^n}) \backslash \mathrm{Div}_C(\mathbb{F}_q)$ *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 given with u-polynomial $u \in \mathbb{F}_{q^n}[x]$ irreducible. Then $\mathrm{Tr}(D)$ has u-polynomial $N(u) = uu^{\varphi} \cdots u^{\varphi^{n-1}}$, where all the u^{φ^j} are irreducible over $\mathbb{F}_{q^n}$. Hence any

D' with $\mathrm{Tr}(D') = \mathrm{Tr}(D)$ has to have as u-polynomial one of the u^{φ^j}, and therefore $D' = \varphi^j(D)$ for some $j \in \{0, \dots, n-1\}$. Conversely, $\mathrm{Tr}(\varphi^j(D)) = \mathrm{Tr}(D)$ for all j.

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

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

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

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

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

We are interested in the kernel of this map.

Definition 2.3. Let n be a prime number. Then the *trace zero subgroup* of $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ is

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

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]. One of the motivations for using the trace zero subgroup is that it allows in principle to reduce the key length. In this paper, we show how to do this for any g and any prime n. Moreover, addition in the trace zero subgroup may be sped up considerably by using the Frobenius endomorphism. The problem of how to speed up the arithmetic in this group was studied by Lange (see [Lan01, Lan04]) and by Avanzi-Cesena (see [AC07, Ces08]), following an approach similar to the one used for Koblitz curves (see [Kob91]) and later applied to GLV-GLS curves (see [GLV01, GLS11]). Furthermore, Rubin and Silverberg showed in [RS09] that pairings defined on higher dimensional abelian varieties, such as the trace zero variety, yield exceptionally high security parameters for some values of n and g. Finally, the trace zero subgroup captures the hardness of the DLP over extension fields. More specifically, the DLP in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ is as hard as the DLP in T_n.

Proposition 2.4. *We have a short exact sequence*

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

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

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

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

Remark 2.5. 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 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 2.6. As a consequence of the exact sequence in Proposition 2.4 we obtain

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

The Hasse-Weil Theorem states that

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

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

The goal of this paper is finding an optimal-size representation for the elements of the trace zero subgroup. This is a natural question, which has been investigated in previous works both for elliptic and hyperelliptic curves, as well as in the analogous case of torus-based cryptography. It is also stated as an open problem in the conclusions of [AC07].

Definition 2.7. A *representation of size ℓ* for the elements of a finite set G is an injection

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

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

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

Remark 2.8. Since the elements of $\mathbb{F}_q$ can be represented via binary strings of length $\log_2 q$, an optimal representation for a set G with $|G| \approx q^m$ can be given via $\mathcal{R} : G \longrightarrow \mathbb{F}_q^m$.

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

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

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

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

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

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

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

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

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

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

The polynomial u contains all the information about the x-coordinates of the points P_i in the reduced representation of $D = P_1 + \ldots + P_r - r\mathcal{O}$, but not about the signs of the corresponding y-coordinates. As before, one can either use g extra bits to store these signs (see Hess-Seroussi-Smart [HSS01]), or one can work with classes $\{[w^{i_1}(P_1) + w^{i_2}(P_2) + \ldots + w^{i_r}(P_r) - r\mathcal{O}] \mid i_j \in \{0,1\}\}$, thus identifying up to 2^g elements of $\operatorname{Pic}^0_C(\mathbb{F}_q)$. For small g, these classes are not too large. A different representation for the elements of $\operatorname{Pic}^0_C(\mathbb{F}_q)$ of size $g \log_2 q + g$ is given by Stahlke [Sta04].

Remark 2.11. Since $T_n \subset \operatorname{Pic}^0_C(\mathbb{F}_{q^n})$, we may use the representation of Example 2.10 for points of the trace zero subgroup. However this is not optimal, since $\log_2 |T_n| \approx (n-1)\log_2 q \not\approx n \log_2 q$.

3. An optimal representation for the trace zero variety via rational functions

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

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

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

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

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

In particular, if $D = (X,Y) - \mathcal{O}$, then $h_D = x - X$, and the x-coordinate of the points of T_2 yields an optimal representation (see [GM14, Proposition 2]). This statement can be generalized to higher genus curves when $n = 2$: The u-polynomial of the Mumford representation yields an optimal size representation.

Proposition 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*

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

Therefore, the map

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

yields an optimal representation for the elements of T_2, where $\mathbb{F}_q[x]_{\leq g}$ denotes the polynomials of degree smaller than or equal to g with coefficients in $\mathbb{F}_q$. Hence we can represent the elements of T_2 via $g \log_2 q + 1$ bits.

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

Conversely, let D be a divisor with Mumford representation $[u,v] \in \mathrm{Pic}^0_C(\mathbb{F}_{q^2})$. If $u \in \mathbb{F}_q[x]$ and $v^\varphi = -v$, then $\varphi([u,v]) = [u,-v]$. Hence $D = w(\varphi(D))$, so $[D + \varphi(D)] = 0$ and $[D] \in T_2$. This concludes the proof of equality (1).

Equality (1) yields a simple, optimal representation for $[D] \in T_2$, as given by the map $\mathcal{R}$. Assume that D has Mumford representation $[u,v]$, with u monic of degree $r \leq g$. Then point compression is for free, and the representation consists of the g coefficients of $1, \ldots, x^{g-1}$ in u, together with an extra bit which indicates whether $\deg u = g$ or not. This representation has length $g \log_2 q + 1$, and it identifies $[D]$ and its conjugate $[w(D)] = [\varphi(D)]$. □

Remark 3.2. Notice that if $g = 1$ and $D = (X,Y) - \mathcal{O}$, then $u(x) = x - X$ is a vertical line, from which (X,Y) can be recovered up to sign. Since $u(x)$ is determined by the coefficient X, the representation consists only of the x-coordinate of the point. Moreover, we do not need to include the extra bit in the representation, since we always have $r = 1$ for $D \neq 0$. Hence this representation coincides with the trivial optimal representation from [GM14, Proposition 2].

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

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

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

Theorem 3.3. *Let $D = P_1 + \ldots + P_r - r\mathcal{O}$ be a reduced divisor such that $[D] \in T_n$, and let $h_D \in \mathbb{F}_q(C)$ be a function such that* $\mathrm{div}(h_D) = \mathrm{Tr}(D)$. *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, $H_D = N(u)$.*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Write $\mathrm{Tr}(D)=\mathrm{Tr}(D_1)+\ldots+\mathrm{Tr}(D_t)=\mathrm{Tr}(F)$, where $\mathrm{Tr}(D_i)\in\mathrm{Div}_C(\mathbb{F}_q)$ are prime divisors by Lemma 2.2 (ii). By Lemma 2.2 (i), $\mathrm{Tr}^{-1}(\mathrm{Tr}(D_i))=\{D_i,\varphi(D_i),\ldots,\varphi^{n-1}(D_i)\}$ for all i, hence $F=\varphi^{j_1}(D_1)+\ldots+\varphi^{j_t}(D_t)$ for some $j_1,\ldots,j_t\in\{0,\ldots,n-1\}$. The number of such F is $n^t\le n^g$.

(vi) We have $h_{D,2}(x)\equiv 0 \bmod u_i$ if and only if $h_D(x,y)\equiv h_{D,1}(x)\equiv h_{w(D)}(x,y) \bmod u_i$. Since $D_i\le\mathrm{Tr}(D)$, this is also equivalent to $w(D_i)\le\mathrm{Tr}(D)$. Since D_i is prime, $w(D_i)$ is also prime and $w(D_i)\le\mathrm{Tr}(D)$ 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 2.2 (i).

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

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

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

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

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

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

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

Remark 3.4. The results of Theorem 3.3 may be generalized to elliptic and hyperelliptic curves over fields of characteristic 2 by defining $H_D=h_D(h_D\circ w)$. It is not hard to check that we obtain a function h_D with the same properties as in (i)–(vii). Special caution needs to be used in adapting (vi). Since the existence of this function is the basis for the representation we propose in this paper, all our results and methods work, with the necessary adjustments, over a finite field of any characteristic.

Let $[D]\in T_n\setminus\{0\}$. One of the consequences of Theorem 3.3 is that, depending on the parity of r and n, we know the exact degree of either $h_{D,1}$ or $h_{D,2}$. By making the appropriate one monic, we can represent h_D by $(n-1)g$ elements of $\mathbb{F}_q$, namely the coefficients of $h_{D,1}$ and $h_{D,2}$, plus one bit which indicates whether $r=g$. This is an optimal representation, since

$(n-1)g\log_2 q + 1 \approx (n-1)g\log_2 q \approx \log_2 |T_n|$. See Remark 3.12 for an alternative approach that does not require storing the extra bit δ.

Corollary 3.5. *Under the assumptions and following the notation of Theorem 3.3, we represent a reduced divisor $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ of degree zero such that $[D] \in T_n$ as follows.*
Let $[u,v]$ be the Mumford representation of D, $u(x) = u_0 + u_1x + \ldots + u_{r-1}x^{r-1} + x^r$, $r = \deg u$. Set $d_1 = \lfloor \frac{ng}{2} \rfloor$ and $d_2 = \left\lfloor \frac{(n-2)g-1}{2} \right\rfloor$. Let $h_{D,1} = \gamma_{d_1}x^{d_1} + \ldots + \gamma_1 x + \gamma_0$, $h_{D,2} = \beta_{d_2}x^{d_2} + \beta_{d_2-1}x^{d_2-1} + \ldots + \beta_1 x + \beta_0$, where $h_{D,1}$ is monic if nr is even, and $h_{D,2}$ is monic if nr is odd. If $r = g$ let $\delta = 1$, else let $\delta = 0$. Define:

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

Then $\mathcal{R}$ is an optimal representation for the non-zero elements of T_n. This representation identifies at most n^g elements, as stated in Theorem 3.3 (v).

Proof. If $n = 2$ the statement follows from Proposition 3.1. For $n \geq 3$, the statement is a direct consequence of the proof of Theorem 3.3 (iii) and (iv). Observe that if $r < g$, then
$$\deg h_{D,1} = \left\lfloor \frac{rn}{2} \right\rfloor \leq \frac{n(g-1)-1}{2} = \frac{ng}{2} - \frac{n+1}{2} \leq d_1 - 2$$
if g is even and
$$\deg h_{D,2} = \left\lfloor \frac{nr-2g-1}{2} \right\rfloor \leq \frac{n(g-1)-2g-2}{2} = \frac{(n-2)g-1}{2} - \frac{n+1}{2} \leq d_2 - 2$$
if g is odd. In particular, the polynomials $h_{D,1}$ and $h_{D,2}$ can always be represented using the number of coefficients claimed. Moreover, if $r < g$ some of the coefficients that we store are zero. In particular, $\gamma_i = 0$ for $i > \lfloor \frac{rn}{2} \rfloor$ and $\beta_i = 0$ for $i > \lfloor \frac{nr-2g-1}{2} \rfloor$. Since
$$d_1 + d_2 + 1 = \left\lfloor \frac{ng}{2} \right\rfloor + \left\lfloor \frac{(n-2)g-1}{2} \right\rfloor + 1 = (n-1)g,$$
we store the correct number of $\mathbb{F}_q$-coefficients in all cases.

Finally, the representation identifies at most n^g elements by Theorem 3.3 (v). □

Remarks 3.6. (i) Let $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ be a reduced divisor, $D = D_1 + \ldots + D_t$ with $D_i \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ reduced prime divisors. Notice that not all the divisors F of the form $F = \varphi^{j_1}(D_1) + \ldots + \varphi^{j_t}(D_t)$ for some $j_1, \ldots, j_t \in \{0, \ldots, n-1\}$ are reduced. E.g., let C be a hyperelliptic curve of genus 2 and let $P \in C(\mathbb{F}_{q^n}) \setminus C(\mathbb{F}_q)$ be a point. Then $\varphi(P) \neq P$ and $D = P + w(\varphi(P)) - 2\mathcal{O}$ is a reduced divisor. But a divisor $F = \varphi^{j_1}(P) + w(\varphi^{j_2}(P))$ is reduced if and only if $j_1 \neq j_2$.

Because of this, when decompressing $\mathcal{R}([D])$ one needs to discard all the divisors classes $[F] \in T_n$ which have $\mathrm{Tr}(F) = \mathrm{Tr}(D)$, but F is not a reduced divisor. E.g., in the previous example we would discard the elements $F = \varphi^j(P) + w(\varphi^j(P)) \sim 0$. In Algorithm 4, for a given $\alpha = \mathcal{R}([D])$ we recover one reduced $F \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ such that $\mathcal{R}([F]) = \alpha$. Because of what we just discussed, F identifies the equivalence class of D.

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

Remark 3.7. If only the u-polynomial of the Mumford representation of $[D]$ is available (e.g. when using the representation for elements of $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$ which we discuss in Example 2.10), one can still compute $H_D = u^{1+\varphi+\ldots+\varphi^{n-1}} \in \mathbb{F}_q[x]$. From $H_D = h_{D,1}^2 - fh_{D,2}^2$, the polynomials $h_{D,1}, h_{D,2}$ may be recovered up to some indeterminacy by solving a linear system in their (unknown) coefficients. If $u = \prod u_i$ where u_i are irreducible over $\mathbb{F}_{q^n}$, then the indeterminacy comes from the fact that for each i we cannot distinguish between u_i and its Frobenius conjugates (since we only know the norm of u_i) and we do not know the sign of v_i. Hence representing a $[D] \in T_n$ via a pair of polynomials $h_{D,1}, h_{D,2}$ obtained as above still has optimal size, but it identifies up to $(2n)^g$ elements. In practice however, in order to compress it suffices to find one solution of the linear system.

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

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

Lemma 3.8. *Let $D = [u, v]$ be a divisor on C, and let $D_i = \varphi^i(D)$ for $i \geq 0$. Let $h^{(1)} = u$ as a function on C, and define recursively the functions*

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

where a is given by Cantor's algorithm according to

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

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

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

If $[D] \in T_n$, then

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

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

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

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

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

as claimed. □

The functions $h^{(i)}$ of Lemma 3.8 can be computed using a Miller-style algorithm, where at each step we apply Cantor's algorithm to compute the function a. When $[D] \in T_n$, then the $(n-1)$-th iteration $h^{(n-1)}$ agrees with the desired function h_D. Hence h_D can easily be computed with a Miller-style algorithm, as in Algorithm 2.

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:     $(R, a) \leftarrow \text{Cantor}(R, \varphi^i(R)), h \leftarrow h \cdot h^{\varphi^i} \cdot a^{-1}, Q \leftarrow \varphi^i(Q), i \leftarrow 2i$
4:     if $n_j = 1$ then
5:         $(R, a) \leftarrow \text{Cantor}(R, Q), h \leftarrow h \cdot u^{\varphi^i} \cdot a^{-1}, Q \leftarrow \varphi(Q), i \leftarrow i + 1$
6:     end if
7: end for
8: return $h$
```

Theorem 3.9. *Algorithm 2 computes h_D correctly. Computing h_D has a complexity of $O(g^4 \log n + g^3 n + g^{\log 3} 3^{\log n})$ operations in $\mathbb{F}_{q^n}$.*

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

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

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

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

Finally, by Lemma 3.8

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

In an addition step, we compute

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

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

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

Finally

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

according to Lemma 3.8. So the output of Algorithm 2 is $h^{(n-1)}$, a function with divisor $\mathrm{Tr}(D)$, as desired.

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

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

We assume that the input divisors to Cantor's algorithm have u-polynomials of degree g, and that they are coprime (this is true generically). Then, at the end of the composition (after line 3 in Algorithm 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. Then, following through the algorithm, one can easily check that

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

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

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

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

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

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

and hence

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

where the divisions are exact, since the results must be polynomials. The most expensive multiplications involved are those by h'_1 and h'_2, since those have the largest degree, namely about g^2+ig. Using Karatsuba multiplication, we can thus compute the numerators and denominators above in a total of $O((g^2+ig)^{\log 3})$ operations. The two long divisions take $O(g^3(i+g))$ each, since both numerators have degree in the order of $g(i+g)$, and both denominators have degree in the order of g^2. Hence the entire computation of $h^{(2i)}$ takes $O(g^4+g^3i+(gi)^{\log 3})$ operations.

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

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

as claimed. □

Remark 3.10. It is also possible to determine h_D by solving a linear system of size about $gn\times gn$ for the coefficients. Using standard Gaussian elimination techniques, this has complexity $O((gn)^3)$. This is larger in n but smaller in g than the complexity of Algorithm 2. Therefore, the linear algebra method is preferable when n is small and g is large.

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 $n\geq 3$, since in the case $n=2$ the representation consists of $u(x)$ (see Proposition 3.1, Theorem 3.3 (iv), and Corollary 3.5).

The compression algorithm follows immediately from the results of the previous section. The strategy of the decompression algorithm is as follows. From the input $\alpha=\text{Compress}(D)$, we recompute $h_{D,1}$ and $h_{D,2}$, and then H_D. Then we factor H_D in order to obtain the u-polynomials of (one Frobenius conjugate of each of) the $\mathbb{F}_{q^n}$-rational prime divisors in D. This is consistent with the fact that $\text{Tr}(D)$ only contains information about the conjugacy classes of these prime divisors. Afterwards, we compute the corresponding v-polynomial for each u-polynomial. In this way, if $D=D_1+\ldots+D_t$ is the decomposition of D as a sum of $\mathbb{F}_{q^n}$-rational prime divisors, we recover an $\mathbb{F}_{q^n}$-rational prime divisor $D'_1,\ldots,D'_t$, where each D'_i is one of the Frobenius conjugates of D_i. The divisor $D'_1+\ldots+D'_t$ corresponds to the class $\mathcal{R}^{-1}(\alpha)$, since every $F\in\mathcal{R}^{-1}(\alpha)$ is of the form $\varphi^{j_1}(D'_1)+\ldots+\varphi^{j_t}(D'_t)$ for some $j_1,\ldots,j_t\in\{0,\ldots,n-1\}$ (see Theorem 3.3 (v)). Since we know that α is the representation of a reduced divisor, we compute a reduced representative $D'_1+\ldots+D'_t$ of the class $\mathcal{R}^{-1}(\alpha)$ (see also Remark 3.6).

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

Theorem 3.11. *(i) Compression Algorithm 3 computes a unique optimal representation of a given $[D]\in T_n$, for D a reduced divisor. It takes $O(g^4\log n+g^3n+g^{\log 3}3^{\log n})$ operations in $\mathbb{F}_{q^n}$.*

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

(iii) Decompression Algorithm 4 takes $O((ng)^{1+\log 3}\log(q^nng))$ operations in $\mathbb{F}_{q^n}$.

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 3.9.

(*ii*) Let $D=D_1+\ldots+D_t$, where D_i are reduced prime divisors defined over $\mathbb{F}_{q^n}$, possibly not all distinct. The reduced divisors D' such that $[D']\in T_n$ and $\text{Compress}(D)=\text{Compress}(D')$ are exactly those for which $h_{D'}=h_D$. By Theorem 3.3 (v), they are of the form $D'=\varphi^{j_1}(D_1)+$

Algorithm 3 Compression, $n \geq 3$

```
Input: $[D] = [u, v] \in T_n$
Output: Representation $(\alpha_0, \dots, \alpha_{(n-1)g}) \in \mathbb{F}_q{}^{(n-1)g} \times \mathbb{F}_2$ 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)
 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:     $h_{D,1} \leftarrow h_{D,1}/\operatorname{lc}(h_{D,1})$   ▷ 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:     $h_{D,2} \leftarrow h_{D,2}/\operatorname{lc}(h_{D,1})$   ▷ 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:     $h_{D,1} \leftarrow h_{D,1}/\operatorname{lc}(h_{D,2})$   ▷ 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:     $h_{D,2} \leftarrow h_{D,2}/\operatorname{lc}(h_{D,2})$   ▷ 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 $(\beta_0, \dots, \beta_{d_2}, \gamma_0, \dots, \gamma_{d_1})$
14: else
15:     return $(\gamma_0, \dots, \gamma_{d_1}, \beta_0, \dots, \beta_{d_2})$
16: end if
```

$\ldots + \varphi^{j_t}(D_t)$ for some $0 \leq j_1, \dots, j_t \leq n-1$, and we will show that Algorithm 4 computes such a divisor D' for some $j_1, \dots, j_t$.

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

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

where $U_i \in \mathbb{F}_q[x]$ are irreducible and $U_i \neq U_j$ if $i \neq j$, $m \leq t$. 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 $\operatorname{Tr}(D_i) = nD_i$. Notice that by Lemma 2.2 (ii) U_i is an $\mathbb{F}_q[x]$-irreducible factor of $H_D(x)$ independently of whether $u_i \in \mathbb{F}_q[x]$ or not. Notice moreover that $u_i \in \mathbb{F}_q[x]$ if and only if U_i is irreducible in $\mathbb{F}_{q^n}[x]$. Conversely, U_i is reducible in $\mathbb{F}_{q^n}[x]$ if and only if $u_i \in \mathbb{F}_{q^n}[x]$ is one of its irreducible factors. Summarizing, each D_i corresponds exactly to a set of n $\mathbb{F}_{q^n}[x]$-irreducible factors of H_D, and these factors can be correctly grouped by first computing the $\mathbb{F}_q[x]$-factorization of $H_D = N(u)$.

Fix $i \in \{1, \dots, m\}$ and let $U(x)$ be an $\mathbb{F}_{q^n}[x]$-irreducible factor of $U_i(x)$, i.e., $U(x)$ is a Frobenius conjugate of $u_i(x)$. We now discuss: how to compute the corresponding $V(x)$, so that $[U, V]$ is the Mumford representation of some Frobenius conjugate of D_i, and how to determine the cardinality m of the set of $j \in \{1, \dots, t\}$ such that $D_j \leq D$ is a Frobenius conjugate of D_i, respectively the cardinality ℓ of the set of $j \in \{1, \dots, t\}$ such that $D_j \leq D$ is a Frobenius conjugate of $w(D_i)$. If $D_i \neq w(D_i)$ we have that $H_D = N(u)$ is divisible by $U_i^m U_i^\ell$ and by no higher power of U_i by Theorem 3.3 (ii), hence $e_i = m + \ell$. If $D_i = w(D_i)$ by definition $m = \ell$ and $H_D = N(u)$ is divisible by U_i^m and by no higher power of U_i by Theorem 3.3 (ii), hence $m = e_i$.

By Theorem 3.3 (vi), if $U \nmid h_{D,2}$ then $\ell = 0$ and no Frobenius conjugate of $w(D_i)$ appears among the prime divisors $D_j \leq D$. Moreover, there exist polynomials $k(x), l(x) \in \mathbb{F}_{q^n}[x]$ such that $k(x)h_{D,2} = 1 + l(x)U(x)$. Hence $k(x)(h_{D,1}(x) + y h_{D,2}(x)) \equiv y + k(x)h_{D,1} \bmod U$. Since $h_{D,1} + y h_{D,2} \equiv 0 \bmod (U, y - V)$, then $y - V$ divides $y + k(x)h_{D,1} \bmod U$. It follows that

Algorithm 4 Decompression, $n \geq 3$

```
Input: $(\alpha_0, \dots, \alpha_{(n-1)g}) \in {\mathbb{F}_q}^{(n-1)g} \times \mathbb{F}_2$
Output: one reduced $D \in \mathrm{Div}_C(\mathbb{F}_{q^n})$ such that $[D] \in T_n$ has representation $(\alpha_0, \dots, \alpha_{(n-1)g})$
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:     $h_{D,1}(x) \leftarrow \alpha_{(n-1)g} x^{d_1} + \dots + \alpha_{d_2+2} x + \alpha_{d_2+1}$
5:     $h_{D,2}(x) \leftarrow \alpha_{d_2} x^{d_2} + \alpha_{d_2-1} x^{d_2-1} + \dots + \alpha_1 x + \alpha_0$
6: else
7:     $h_{D,1}(x) \leftarrow \alpha_{d_1} x^{d_1} + \dots + \alpha_1 x + \alpha_0$
8:     $h_{D,2}(x) \leftarrow \alpha_{(n-1)g} x^{d_2} + \dots + \alpha_{d_1+2} x + \alpha_{d_1+1}$
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,
    $e_i \in \{1, \dots, gn\}$
12: $L \leftarrow$ empty list
13: for $i = 1, \dots, m$ do
14:     if $U_i(x)$ is irreducible over $\mathbb{F}_{q^n}$ then        ▷ $U_i$ 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                                                        ▷ $h_{D,2}(x) \equiv 0 \bmod U(x)$
22:         if $f(x) \equiv 0 \bmod U(x)$ then                       ▷ $V(x) = 0$ and $D_i = w(D_i)$
23:             append $[U(x), 0], [U(x)^{\varphi}, 0], \dots, [U(x)^{\varphi^{e_i-1}}, 0]$ to $L$
24:         else                                                    ▷ $V(x) \neq 0$ and $D_i \neq w(D_i)$
25:             compute $\ell, h'_D$ such that $h_D = U_i(x)^{\ell} h'_D$ and $U_i(x) \nmid h'_D$
26:             if $\ell < e_i/2$ then
27:                 $V(x) \leftarrow -h'_{D,1}(x) h'_{D,2}(x)^{-1} \bmod U(x)$
28:                 append $[U(x), V(x)]$ to $L, e_i - \ell$ times
29:                 append $[U(x)^{\varphi}, -V(x)^{\varphi}]$ to $L, \ell$ times
30:             else                                                ▷ $\ell = e_i/2$
31:                 $V(x) \leftarrow \sqrt{f(x)} \bmod U(x)$
32:                 append $[U(x), V(x)], [U(x)^{\varphi}, -V(x)^{\varphi}]$ to $L, \ell$ times
33:             end if
34:         end if
35:     end if
36: end for                                                         ▷ Notation: $L = [D_1, \dots, D_t]$
37: return $D = D_1 + \dots + D_t$
```

$y - V \equiv y + k(x) h_{D,1} \bmod U$, hence

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

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

If $U \mid h_{D,2}$, it follows from Theorem 3.3 (vi) that $w(D_i) = \varphi^j(D_k)$ for some $0 \leq j \leq n-1$ and $1 \leq k \leq t$. In other words, both $[u_i, v_i] = D_i$ and some Frobenius conjugate of $[u_i, -v_i] = w(D_i) = \varphi^j(D_k)$ appear in D. We distinguish the cases

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

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

Case (b) is treated in lines 25–33 of the algorithm. In this case $i \neq k$ by Theorem 3.3 (vi) since $n \neq 2$. By Theorem 3.3 (vii), $\mathrm{Tr}(D) = m\,\mathrm{Tr}(D_i) + \ell\,\mathrm{Tr}(w(D_i)) + \mathrm{Tr}(G)$ where $\mathrm{Tr}(D_i), \mathrm{Tr}(w(D_i)) \not\leq \mathrm{Tr}(G)$ and $s := \min\{m, \ell\}$ may be computed as the exponent for which $U_i^s \mid h_D$ and $U_i^{s+1} \nmid h_D$. Let $h_D = U_i^s h'_D$, $h'_D = h'_{D,1} + y h'_{D,2}$. Notice that $U, U_i \nmid h'_{D,2}$, since $\mathrm{Tr}(D_i) + \mathrm{Tr}(w(D_i)) \not\leq \mathrm{div}(h'_D)$. Then: $s = m = \ell = e_i/2$ if and only if $\mathrm{Tr}(D_i), \mathrm{Tr}(w(D_i)) \not\leq \mathrm{div}(h'_D)$ if and only if $U \nmid V^2 - f$, where $V = -h'_{D,1} {h'_{D,2}}^{-1} \bmod U$. In this case, we can let $V = \sqrt{f} \bmod U$ and D contains exactly $e_i/2$ Frobenius conjugates of $[U, V]$ and $e_i/2$ Frobenius conjugates of $[U, -V]$. If instead $s = \ell < m$, then $h_D = U_i^\ell h'_D$ and $\mathrm{div}(h'_D) \geq \mathrm{Tr}(D_i)$, $\mathrm{div}(h'_D) \ngeq \mathrm{Tr}(w(D_i))$. Therefore, $U \nmid h'_{D,2}$ and V can be computed as $V = -h'_{D,1} {h'_{D,2}}^{-1} \bmod U$. In this case, D contains exactly $m = e_i - \ell$ Frobenius conjugates of $[U, V]$ and ℓ Frobenius conjugates of $[U, -V]$.

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

Finally, we show that the D' returned by Algorithm 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 case (a) we have $D_i = w(D_i)$, and we need to check that $D'_i, \varphi(D'_i), \ldots, \varphi^{e_i - 1}(D'_i)$ are distinct, where $D'_i = [U, 0]$. In particular, we check that $e_i < n$. But if D'_i, hence D_i, were $\mathbb{F}_q$-rational or $e_i > n$, then D would not be reduced. In case (b) we have $D_i \neq w(D_i)$ and hence $w(\varphi(D_i)) \neq \varphi(D_i)$, so we may add several times $D'_i = [U, V]$ and $w(\varphi(D'_i)) = [U^\varphi, -V^\varphi]$. Furthermore, we have $D'_i \neq \varphi(D'_i)$. Indeed, if $D'_i = \varphi(D'_i)$ then it follows that D'_i, D_i and D_k are $\mathbb{F}_q$-rational. Hence $D_i = w(D_k)$, a contradiction to the fact that D is reduced, since $i \neq k$ in this case.

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

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

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

Remark 3.12. The representation computed by Algorithm 3 has size $(n-1)g\log_2 q + 1$. If one chooses to work only with divisors of the form $D = P_1 + \ldots + P_g - g\mathcal{O}$, then the last bit may be dropped and we have a representation of size $(n-1)g\log_2 q$. Divisor classes whose reduced representative has this form constitute the majority of the elements of T_n. Moreover, there are cases in which the trace zero subgroup consists only of divisor classes represented by reduced divisors of this shape. This is the case for elliptic curves, where $r = 1$ if $D \neq 0$. Moreover, Lange [Lan04, Theorem 2.2] proves that for $g = 2$ and $n = 3$, all nontrivial elements of T_3 are represented by reduced divisors with $r = 2 = g$.

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

(i) perform the operation in the compressed coordinates, or to
(ii) 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, FHLS13]. Timings typically ignore the additional cost for point decompression, but there is strong evidence that on a large class of elliptic curves the second approach is faster. 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 points of a trace zero subgroup: Decompress the point, perform the operation in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$, and compress the result. Since our compression and decompression algorithms are very efficient, this adds only little overhead. Moreover, scalar multiplication is considerably more efficient for trace zero points than for general points in $\mathrm{Pic}^0_C(\mathbb{F}_{q^n})$, due to 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. As we will see, it is also much easier to find a rational function with a given principal divisor. For all these reasons, the results and methods from Section 3 can be simplified and made explicit for elliptic curves.

Throughout this section, 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. In this section we consider only $n \geq 3$, since the case $n = 2$ is rather trivial, as explained at the beginning of Section 3. In this setting, we have an analogous but more explicit version of Theorem 3.3.

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. We follow the usual convention that the line passing through P with multiplicity two is the tangent line to the curve at P. Let $v_i(x, y) = 0, i = 1, \ldots, n-3$, be the equation of the vertical line passing through the point $P_0 \oplus \ldots \oplus P_i$.

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 ℓ_j and v_j are the lines defined in Notation 4.1. Then:

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

Proof. *(i)* We have $\operatorname{div}(\ell_i) = (P_0 \oplus \ldots \oplus P_{i-1}) + P_i + w(P_0 \oplus \ldots \oplus P_i) - 3\mathcal{O}$ for $i \in \{1, \ldots, n-2\}$ and $\operatorname{div}(v_i) = (P_0 \oplus \ldots \oplus P_i) + w(P_0 \oplus \ldots \oplus P_i) - 2\mathcal{O}$ for $i \in \{1, \ldots, n-3\}$. Now we compute

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

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

(ii), (iii), (iv), (v), (vi) follow directly from Theorem 3.3.

(vii) Since $h_{P,2}$ is defined over $\mathbb{F}_q$, it suffices to show $h_{P,2}(X) \neq 0$ for the x-coordinate of P. By (ii) $h_{P,2}$ is not the zero polynomial, since otherwise $\deg H_P = 2 \deg h_{P,1} \leq n-1$. If $n = 3$ then $\deg h_{P,2} = 0$ by (iv), hence $h_{P,2}(X)$ is a non-zero constant for every X. Now suppose $n > 3$ and $h_{P,2}(X) = 0$, then it follows from Theorem 3.3 (vi) that $w(P) = \varphi^j(P)$ for some j. If $j \neq 0$, then $X \in \mathbb{F}_{q^n} \cap \mathbb{F}_{q^j} = \mathbb{F}_q$ and $Y \in \mathbb{F}_{q^n} \cap \mathbb{F}_{q^{2j}} = \mathbb{F}_q$. Hence $P = w(P)$, i.e. P and all its Frobenius conjugates are 2-torsion points. On the other hand, $P \in T_n$ implies

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

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

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

Corollary 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*

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

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

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

We detail in Algorithms 5 and 6 the simplified compression and decompression algorithms for elliptic curves.

Finally, we discuss how to compute h_P for a given $P \in T_n$. Explicit formulas can be computed in the special cases $n = 3, 5$. We do this in the next section. For general n, a straightforward computation of h_P is possible, since we have an explicit formula given in terms of lines whose equations we know. Such a computation can be made more efficient by employing the usual

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

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 $P = (X, Y)$
```

divide and 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 (for which we do not have explicit formulas) a straightforward computation using a divide and conquer approach seems preferable.

In Algorithm 7 we give a Miller-style algorithm to compute h_P. We denote by $\ell_{P,Q}$ the line through the points P and Q, and by v_P the vertical line through P. All computations are done with functions on E, i.e. in $\mathbb{F}_{q^n}(E)$.

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

Corollary 4.4. (*i*) *The execution of Algorithm 7, and therefore also of compression Algorithm 5, requires $O(3^{\log_2 n})$ operations in $\mathbb{F}_{q^n}$.*

(*ii*) Decompress(Compress(P)) *is one of the Frobenius conjugates of P. The complexity of decompression Algorithm 6 is $O(n^{\log 3+1} \log n \log(nq))$ operations in $\mathbb{F}_{q^n}$.*

Proof. (*i*) See Theorem 3.9.

(*ii*) Theorem 3.11 (vi) would give a complexity of $O(n^{2+\log_2 3}\log(nq))$ operations in $\mathbb{F}_{q^n}$. However, the situation here is simpler, since we know that H_D must split into linear factors over $\mathbb{F}_{q^n}$. Therefore, we apply the root finding algorithm of [GvzG99, Algorithm 14.15], which has a better complexity of $O(n^{\log 3+1}\log n\log(nq))$ operations in $\mathbb{F}_{q^n}$. □

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. Explicit equations

For the simple cases, we give explicit equations for compression and decompression. In addition to making the computation more efficient, they allow us to perform precise operation counts, and thus to compare our method to the other existing compression methods in Section 6.

5.1. Explicit equations for $g = 1, n = 3$.

We now study how to best compute the compression and decompression of elliptic curve points when $n = 3$ and give explicit formulas. This case is particularly simple, since $h_P = \ell_1$ is just a line through the points $P, \varphi(P), \varphi^2(P)$.

For the sake of concreteness, we assume that $\mathbb{F}_q$ does not have characteristic 2 or 3 and that E is given by an equation in short Weierstrass form

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

Formulas and operation counts can easily be adjusted to the other cases. It will also be useful to choose a basis of the field extension $\mathbb{F}_{q^3}|\mathbb{F}_q$. For simplicity, we also assume that $3 \mid q-1$ and write $\mathbb{F}_{q^3} = \mathbb{F}_q[\zeta]/(\zeta^3-\mu)$ as a Kummer extension, where $\mu \in \mathbb{F}_q$ is not a third power. Then $1, \zeta, \zeta^2$ is a basis of $\mathbb{F}_{q^3}|\mathbb{F}_q$. It is highly likely that there exists a suitable μ of small size, see [Lan04, Section 3.1].

Remark 5.1. When required to work with a field extension where $3 \nmid q-1$, one may choose 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

$$h_P = y - \frac{Y^q - Y}{X^q - X}x + \frac{Y^q - Y}{X^q - X}X - Y = yh_{P,2}(x) + h_{P,1}(x).$$

Therefore, we have $h_{P,2}(x) = 1$ and $h_{P,1}(x) = \gamma_1 x + \gamma_0$, where

$$\begin{aligned}\gamma_1 &= -\frac{Y^q - Y}{X^q - X}\\ \gamma_0 &= -\gamma_1 X - Y,\end{aligned}$$

and Compress$(P) = (\gamma_0, \gamma_1)$.

Since γ_0, γ_1 are in $\mathbb{F}_q$, it is more efficient to carry out the entire computation in $\mathbb{F}_q$. We count squarings (S), multiplications (M), and inversions (I) in $\mathbb{F}_q$. We do not count multiplication by constants. Hence, applying powers of φ is free. Using the basis $1, \zeta, \zeta^2$ of $\mathbb{F}_{q^3}|\mathbb{F}_q$, we write

$$\begin{aligned}X &= X_0 + X_1\zeta + X_2\zeta^2\\ Y &= Y_0 + Y_1\zeta + Y_2\zeta^2.\end{aligned} \tag{3}$$

Then γ_0 and γ_1 can be computed directly from $X_0, X_1, X_2, Y_0, Y_1, Y_2$ over $\mathbb{F}_q$ as

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

where

$$\begin{aligned}c_1 &= 1 - \mu^{(q-1)/3} \\ c_2 &= \mu^{1+(q-1)/3} - \mu = -\mu c_1\end{aligned}$$

are constants and can be precomputed during the setup phase of the algorithm.

When $P \in E(\mathbb{F}_q)$, the line ℓ_1 is a tangent and we have

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

Notice that such points are in $E[3](\mathbb{F}_q)$ and therefore very few. In the general case $P \notin E(\mathbb{F}_q)$ compression takes 2S+6M+1I in $\mathbb{F}_q$.

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

$$H_P(x) = x^3 - S_1 x^2 + S_2 x - S_3$$

where

$$\begin{aligned}S_1 &= \gamma_1^2 \\ S_2 &= A - 2\gamma_0\gamma_1 \\ S_3 &= \gamma_0^2 - B.\end{aligned}$$

Computing the coefficients of H_P therefore takes 2S+1M in $\mathbb{F}_q$. Since the roots of this polynomial are X, X^q, X^{q^2}, and using (3), we get

$$\begin{aligned}S_1 &= X + X^q + X^{q^2} &&= 3X_0 \\ S_2 &= X^{1+q} + X^{1+q^2} + X^{q+q^2} &&= 3X_0^2 - 3\mu X_1 X_2 \\ S_3 &= X^{1+q+q^2} &&= X_0^3 - 3\mu X_0 X_1 X_2 + \mu X_1^3 + \mu^2 X_2^3.\end{aligned}$$

Hence one can solve the system

$$\begin{aligned}S_1 &= 3x_0 \\ S_2 &= 3x_0^2 - 3\mu x_1 x_2 \\ S_3 &= x_0^3 - 3\mu x_0 x_1 x_2 + \mu x_1^3 + \mu^2 x_2^3\end{aligned}$$

over $\mathbb{F}_q$, to recover (X_0, X_1, X_2). Since the solutions of the system are exactly the Frobenius conjugates of X via (3), 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 [GM14, Section 5]). Notice that, since this system is so simple, this is more efficient than factoring the polynomial H_P over $\mathbb{F}_{q^3}$.

Finally, $Y = -\gamma_1 X - \gamma_0$, so recomputing one y-coordinate takes 1M in $\mathbb{F}_q$, and the other ones can be recovered via the Frobenius map.

In total, decompression takes at most 5S+5M+1I, one square root, and two cube roots in $\mathbb{F}_q$.

5.2. **Explicit equations for $g = 1, n = 5$.** We also give explicit formulas for elliptic curves and field extensions of degree 5. Our experimental results show that these formulas are the fastest way to compress and decompress points, and we use them in our implementation, as discussed in Section 6. Again, we assume that E is given in short Weierstrass form $E : y^2 = x^3 + Ax + B$ over a field of characteristic not equal to 2 or 3.

Compression. Let $P = (X, Y) \in T_5$ and denote by $\lambda_1, \lambda_2, \lambda_3$ the slopes of the lines ℓ_1, ℓ_2, ℓ_3, respectively. We have

$$h_P = \frac{\ell_1\ell_2\ell_3}{v_1v_2} = (\gamma_2x^2 + \gamma_1x + \gamma_0) + y(x + \beta_0),$$

where

$$\begin{array}{rcl}
\gamma_2 & = & -\lambda_1 - \lambda_2 - \lambda_3 \\
\beta_0 & = & -\lambda_2\gamma_2 + \lambda_1\lambda_3 - X^{q^2} \\
\gamma_1 & = & -\lambda_2\beta_0 - \gamma_2X^{q^2} + \lambda_1X + \lambda_3X^{q^3} - Y - Y^{q^2} - Y^{q^3} \\
\gamma_0 & = & \gamma_1(\lambda_2^2 - X^{q^2}) + \gamma_2((X + X^q)(X + X^q - X^{q^2} - 2\lambda_1^2 + \lambda_2^2) + \lambda_1^4 + A + \lambda_1^2X^{q^2}) \\
 & & +\lambda_1\lambda_2\lambda_3(X + X^{q^2} + X^{q^3}) - \lambda_1\lambda_2Y^{q^3} - \lambda_1\lambda_3Y^{q^2} - \lambda_2\lambda_3Y + \lambda_3\lambda_1^2\lambda_2^2 + \lambda_1^3\lambda_2^2 + \lambda_1^2\lambda_2^3.
\end{array}$$

Here, we do arithmetic in $\mathbb{F}_{q^5}$ and therefore count operations in $\mathbb{F}_{q^5}$. Computing $\lambda_1, \lambda_2, \lambda_3$ takes a total of 3M+3I. Then, $\beta_0, \gamma_0, \gamma_1, \gamma_2$ can be computed with a total of 3S+15M. Thus, compression takes a total of 3S+18M+3I in $\mathbb{F}_{q^5}$.

Decompression. We compute

$$\begin{array}{rcl}
S_1 & = & \gamma_2^2 - 2\beta_0 \\
S_2 & = & \beta_0^2 + A - 2\gamma_1\gamma_2 \\
S_3 & = & \gamma_1^2 + 2\gamma_0\gamma_2 - 2A\beta_0 - B \\
S_4 & = & A\beta_0^2 + 2B\beta_0 - 2\gamma_0\gamma_1 \\
S_5 & = & \gamma_0^2 - B\beta_0^2
\end{array}$$

using 4S+3M in $\mathbb{F}_q$. Then we factor the polynomial $H_P(x) = x^5 - S_1x^4 + S_2x^3 - S_3x^2 + S_4x - S_5$, which takes $O(\log_2 q)$ operations in $\mathbb{F}_q$. Finally, recovering Y costs 1S+3M+1I in $\mathbb{F}_{q^5}$.

5.3. **Explicit equations for $g = 2, n = 3$.** 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 divisor classes in T_3 are always represented by reduced divisors of a certain shape.

Theorem 5.2 ([Lan04, Theorem 2.2])**.** *Assume that C has genus 2 and that $2, 3 \nmid |\mathrm{Pic}^0_C(\mathbb{F}_{q^3})|$. Then all non-trivial elements of T_3 are represented by reduced divisors of the form*

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

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

Corollary 5.3. *Assume that C has genus 2 and that $2, 3 \nmid |\mathrm{Pic}^0_C(\mathbb{F}_{q^3})|$. Then all non-trivial elements of T_3 are represented by reduced divisors of the form $D = P_1 + P_2 - 2\mathcal{O} \notin \mathrm{Div}_C(\mathbb{F}_q)$, and one of the following mutually exclusive facts holds:*

(*i*) *$P_1, P_2 \in C(\mathbb{F}_{q^3}) \setminus \{\mathcal{O}\}$ and $P_1 \in \{w(\varphi(P_2)), w(\varphi^2(P_2))\}$,*
(*ii*) *$P_1, P_2 \in C(\mathbb{F}_{q^3}) \setminus \{\mathcal{O}\}$ and $P_1 \neq P_2, \varphi(P_2), \varphi^2(P_2), w(\varphi(P_2)), w(\varphi^2(P_2))$,*
(*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. (ii) By contradiction, assume that $h_{D,2} \equiv 0 \bmod u$. Let $P_j = (X_j, Y_j)$, $j = 1, 2$. $P_j - \mathcal{O} \in \mathrm{Div}_C(\mathbb{F}_{q^3})$ is a reduced prime divisor. Since $h_{D,2}(X_j) = 0$, by Theorem 3.3 (vi) we have $w(P_j) = \varphi^i(P_j)$. Then $X_j \in \mathbb{F}_{q^3} \cap \mathbb{F}_{q^i} = \mathbb{F}_q$ and $Y_j \in \mathbb{F}_{q^3} \cap \mathbb{F}_{q^{2i}} = \mathbb{F}_q$. Hence $D = P_1 + P_2 - 2\mathcal{O} \in \mathrm{Div}_C(\mathbb{F}_q)$, which contradicts Theorem 5.2.

(iii) Let $D = P_1 + P_2 - 2\mathcal{O} \in \mathrm{Div}_C(\mathbb{F}_{q^3}) \setminus \mathrm{Div}_C(\mathbb{F}_q)$, and assume that $P_1 \notin C(\mathbb{F}_{q^3})$. Then also $P_2 \notin C(\mathbb{F}_{q^3})$ and $\varphi^3(D) = D$ forces $P_1 = \varphi^3(P_2)$ and $P_2 = \varphi^3(P_1)$. If $w(D) = \varphi^i(D)$ for some $i = 1, 2$, then either $w(P_1) = \varphi^i(P_1)$ or $w(P_1) = \varphi^{i+3}(P_1)$. Hence $P_1 = (X, Y)$ and $\varphi^j(P_1) = (X^{q^j}, Y^{q^j})$ lie on the same vertical line for some $j \in \{i, i+3\}$, therefore $X = X^{q^j} \in \mathbb{F}_{q^6} \cap \mathbb{F}_{q^j} \subseteq \mathbb{F}_{q^2}$ and $Y = -Y^{q^j} \in \mathbb{F}_{q^6} \cap \mathbb{F}_{q^{2j}} \subseteq \mathbb{F}_{q^2}$. This shows that $D \in \mathrm{Div}_C(\mathbb{F}_{q^2}) \cap \mathrm{Div}_C(\mathbb{F}_{q^3}) = \mathrm{Div}_C(\mathbb{F}_q)$, which contradicts Theorem 5.2. Therefore $w(D) \neq \varphi^i(D)$ for $i = 0, 1, 2$, which by Theorem 3.3 (vi) implies that $u \nmid h_{D,2}$, where $[u,v]$ is the Mumford representation of $[D]$. In particular, $h_{D,2} \neq 0$. □

We assume $2, 3 \nmid |\mathrm{Pic}^0_C(\mathbb{F}_{q^3})|$ throughout this section. Hence we know that the Mumford representation of all non-trivial elements of T_3 has a u-polynomial of degree 2 in $\mathbb{F}_{q^3}[x]$. Furthermore, we assume that the characteristic of $\mathbb{F}_q$ is not equal to 2 or 5. In such a case, a simple transformation yields a curve equation of the shape

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

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

Compression. We consider elements $0 \neq [D] = [u,v] \in T_3$ with $D = P_1 + P_2 - 2\mathcal{O}$ where u and u^φ are coprime. This is true under the additional assumption that the x-coordinates of both P_1, P_2 are not in $\mathbb{F}_q$. In addition we assume that $P_1 \neq w(\varphi(P_2)), w(\varphi^2(P_2))$, which by Corollary 5.3 implies that $h_{D,2}$ is not the zero polynomial. The other special cases can be worked out separately, and we do not treat them here.

Proposition 5.4. *Let $0 \neq [D] = [u,v] \in T_3$ such that $\gcd(u, u^\varphi) = 1$ and $h_{D,2}$ is not the zero polynomial. Let $[U,V]$ be the Mumford representation of the semi-reduced divisor $D + \varphi(D)$. Then*

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

Proof. The divisor $D + \varphi(D)$ is semi-reduced by Corollary 5.3. According to Theorem 3.3 (iii), we have $h_D = h_{D,1} + yh_{D,2}$ with $\deg h_{D,1} = 3$ and $\deg h_{D,2} \leq 0$. Since $h_{D,2} \neq 0$ by assumption, and after multiplication by a constant, we have $h_D = y - \gamma(x)$ where $\gamma \in \mathbb{F}_q[x]$ of degree 3. Now if $D = P_1 + P_2 - 2\mathcal{O}$ with $P_i = (X_i, Y_i)$, then $h_D(X_i^{q^j}, Y_i^{q^j}) = 0$ and hence $\gamma(X_i^{q^j}) = Y_i^{q^j}$ for $i = 1, 2$, $j = 0, 1, 2$. But V is the unique polynomial of degree ≤ 3 with $V(X_i^{q^j}) = Y_i^{q^j}$ for $i = 1, 2, j = 0, 1, 2$, and therefore $\gamma = V$.

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

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

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

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

as claimed. □

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

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

Computing these values in a straight forward way takes 2S+32M+1I in $\mathbb{F}_{q^3}$. This number could probably be optimized by regrouping the terms in a more sophisticated way. This is the total compression cost.

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

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

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

6. Timings and comparison with other representations

Important achievements of this new representation are that it works for any prime n and any genus and can be made practical for large values of n and/or g. Moreover our decompression algorithm allows the unique recovery of one well-defined class of conjugates of the original point. For elliptic curves, such a class consists exactly of the Frobenius conjugates of the original point, and for higher genus curves, classes are as described in Theorem 3.3 (v) and Corollary 3.5. Identifying these conjugates is the natural choice from a mathematical point of view, since it respects the structure of our object and is compatible with scalar multiplication of points.

There are only three other known methods for point compression in trace zero varieties over elliptic curves, namely [Nau99], [Sil05], and [GM14]. While [Nau99] only applies to extension degree 3, [Sil05, GM14] can be made practical for $n = 3, 5$. The approach of [GM14] 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 it 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 [GM14] 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 $g = 1$ and $n = 3, 5$, the time for compression and decompression together is comparable for $n = 3$, and smaller for $n = 5$, than that of [GM14]. 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 [GM14, Sil05, Lan04, Nau99].

We now compare the efficiency of our algorithms with those of [GM14, Sil05, Lan04, Nau99] in more detail. The comparison of our method with that of [GM14] is on the basis of a precise operation count, complexity analysis, and our own Magma implementations. Notice that our programs are straight forward implementations of the methods described here and in [GM14], 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) on a Fujitsu Primergy RX900S1. Our timings are average values for one execution of the algorithm, where averages are computed over 10000 executions with random inputs. 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.

Comparison and Timings for $g = 1, n = 3$. We compare our method with the better method of [GM14] (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 [GM14] requires fewer operations, but not in a significant way. These small differences are obviously not measured accurately in our tests. Our measurements for decompression are more meaningful, however. We compare "full decompression", where one entire point (including the y-coordinate) is recomputed. Here, the method of [GM14] is much slower (roughly a factor 10), due to the necessary square root extraction. This shows one major efficiency advantage of the approach that we follow in this 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 [GM14] 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

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 [GM14]	1M
Full decompression	5S+5M+1I, 1 square root, 2 cube roots
Full decompression [GM14]	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 [GM14]	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 [GM14]	0.01	0.02	0.03	0.04
Full decompression	0.18	0.71	0.89	1.52
Full decompression [GM14]	0.84	7.62	10.62	17.58
Decompression x only	0.15	0.63	0.87	1.40
Decompression x only [GM14]	0.15	0.68	0.87	1.44

expensive than the decompression algorithm in this paper, which does not require polynomial factorization or square root extraction.

In [Sil05], compression is for 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 [GM14, Section 5] for a more detailed discussion of the decompression algorithm from [Sil05].

Comparison and Timings for $g = 1, n = 5$. A similar comparison for extension degree 5 (see Tables 3 and 4) shows that the compression algorithm proposed in this paper is less efficient than that of [GM14], 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 [GM14] 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 apper performs much better than that of [GM14], regardless of whether we include the recovery of the y-coordinate. Notice that we again compare with the best method from [GM14], 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 [GM14, 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 subgoups of approximately 160 bits. From the different values for decompression times (due to the fact that the performance of the polynomial

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 [GM14]	5S+13M in $\mathbb{F}_q$
Full decompression	$O(\log q)$ operations in $\mathbb{F}_q$
Full decompression [GM14]	$O(\log q)$ 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 [GM14]	$O(\log q)$ 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 [GM14]	0.04	0.04	0.05	0.10
Full decompression	0.82	9.39	4.26	10.13
Full decompression [GM14]	5.89	17.90	30.21	63.60
Decompression x only	0.77	9.36	4.01	9.82
Decompression x only [GM14]	5.53	16.48	21.42	45.08

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

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.

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 since this is not the focus of our work, we did not put much effort into producing suitable curves for larger trace zero subgroups. Avanzi-Cesena report in [AC07] that they were able to produce trace zero subgroups of 160 and 190 bits for curves of

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

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

genus 2 over fields of even characteristic by modifying a software package written by Frederik Vercauteren.

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 plugs the values of $u_{12}, u_{11}, u_{10}, u_{02}$ into the system and solves for the two missing values in order to determine a, b, which in turn determine the roots coinciding with u_{01}, u_{00}. The decompression algorithm uses a, b to decide which among the solutions of the system are the coordinates it recovers. The advantage of this algorithm is that it works entirely over $\mathbb{F}_q$. Nevertheless, compression is 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 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).

7. CONCLUSION

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

[FHLS13] A. Faz-Hernández, P. Longa, and A. H. Sánchez, *Efficient and secure algorithms for GLV-based scalar multiplication and their implementation on GLV-GLS curves*, Available at `http://eprint.iacr.org/2013/158`, 2013.

[Fre99] G. Frey, *Applications of arithmetical geometry to cryptographic constructions*, 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.

[GM14] E. Gorla and M. Massierer, *Point compression for the trace zero subgroup over a small degree extension field*, To appear in Des. Codes Cryptogr., Springer, DOI: 10.1007/s10623-014-9921-0, 2014.

[GV05] R. Granger and F. Vercauteren, *On the discrete logarithm problem on algebraic tori*, Advances in Cryptology: Proceedings of CRYPTO '05 (V. Shoup, ed.), LNCS, vol. 3621, Springer, 2005, pp. 66–85.

[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.

[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.

[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.

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@unibas.ch