TORUS-BASED COMPRESSION BY FACTOR 4 AND 6

KORAY KARABINA

ABSTRACT. We extend the torus-based compression technique for cyclotomic subgroups and show how the elements of certain subgroups in characteristic two and three fields can be compressed by a factor of 4 and 6, respectively. Our compression and decompression functions can be computed at a negligible cost. In particular, our techniques lead to very efficient exponentiation algorithms that work with the compressed representations of elements and can be easily incorporated into pairing-based protocols that require exponentiations or products of pairings.

1. INTRODUCTION

It has been an attractive objective in cryptography to reduce bandwidth requirements while not forfeiting security and efficiency. For example, Montgomery's [11] scalar multiplication algorithm for a certain class of elliptic curves defined over odd characteristic fields only involves the x-coordinate of the input point $P = (x, y)$. In addition to its advantage of being able to discard the y-coordinate of P, the algorithm can be implemented based on Lucas chains and provides a built-in resistance against certain side-channel attacks. Later on, Montgomery's idea was improved and generalized to any elliptic curve defined over odd characteristic fields and also to elliptic curves defined over binary fields (see for example [1, 10]). Similarly, there have been several proposals to compress the elements of certain subgroups of the multiplicative groups of certain finite fields, and to compute with the compressed representation of elements [16, 4, 2, 9, 3, 12, 20, 19, 15, 8]. The most notable of these proposals is the XTR cryptosystem [9] which compresses the elements of the order-$(p^2 - p + 1)$ subgroup G of $\mathbb{F}_{p^6}^*$ by a factor of three, and at the same time achieves faster exponentiation in G compared to the previous algorithms that work with the natural representation of elements in $\mathbb{F}_{p^6}$.

The compression methods in the finite field setting fall into two category. They either use a rational parameterization of an algebraic torus [12, 20, 19], or use the trace representation of elements [16, 4, 2, 9, 3, 15, 8]. Even though there is a close relationship between these two methods (see [12]), they have different properties. While the rational parametrization of a torus enjoys the full functionality of the group structure, the trace function is not multiplicative and hence novel techniques are required to adapt fast exponentiation algorithms to work with the trace representation of elements (see for example [17, 7]).

Having briefly mentioned the two different compression approaches in finite fields it is natural to ask the following two questions. First, what is the best possible compression ratio for the elements of a subgroup G of the multiplicative group $\mathbb{F}^*$ of a finite field $\mathbb{F}$, where $\mathbb{F}$ is the minimal field with $G \subset \mathbb{F}^*$? One should of course require the corresponding compression and decompression functions to be efficiently computable, and the decompression of an element to be unique or almost unique. Second, how does the tori-compression method compare to the trace-compression method?

The first question was partially answered in [13] for cyclotomic subgroups of finite fields. More precisely, Rubin and Silverberg observed that for a positive integer k and a prime power q

Key words and phrases. Cyclotomic subgroups, torus-based compression, exponentiation, pairing-based cryptography.

there is an algebraic torus $\mathbb{T}_{q,k}$, a $\varphi(k)$-dimensional algebraic variety over $\mathbb{F}_q$, and its group $\mathbb{T}_{q,k}(\mathbb{F}_q)$ of $\mathbb{F}_q$-rational points is isomorphic to the order-$\Phi_k(q)$ (cyclotomic) subgroup of $\mathbb{F}_{q^k}^*$. Here, $\Phi_k(q)$ is the kth-cyclotomic polynomial evaluated at q, and φ is Euler's totient function. Consequently, one would hope to use only $\varphi(k)$ $\mathbb{F}_q$-elements in order to (uniquely) represent elements of $\mathbb{T}_{q,k}(\mathbb{F}_q)$. For example, when $k = 4$ and $k = 6$, the elements of the order-(q^2+1) subgroup $G_{q,4}$ of $\mathbb{F}_{q^4}^*$ and the elements of the order-(q^2-q+1) subgroup $G_{q,6}$ of $\mathbb{F}_{q^6}^*$ can efficiently be compressed and decompressed by a factor 2 and 3, respectively, attaining the best possible compression ratio $k/\varphi(k)$ (see [12, 13]). Recently, it was shown in [15, 8] that it is possible to further compress (and decompress) the elements of certain proper subgroups G_ℓ of $G_{q,4}$ in characteristic-two fields, and $G_{q,6}$ in characteristic-three fields by an additional factor 2, thereby obtaining compression factors 4 and 6, respectively. These seem to be the optimal compression factors as $|G_\ell| = q \pm \sqrt{2q}+1$ and $|G_\ell| = q \pm \sqrt{3q}+1$ in characteristic two and three, respectively, whereby $|G_\ell| \approx q$. In general, it would be desirable to compress the elements of any order-ℓ subgroup $G_\ell \subsetneq G_{q,k} \subset \mathbb{F}_{q^k}^*$ by a factor $(k \log q)/\log \ell$ in any characteristic.

In this paper, we look for answers to these questions. Our arguments suggest that, the *torus-compression* techniques cannot, in general, be extended to achieve compression factor $(k \log q)/\log \ell$ for proper subgroups G_ℓ of $G_{q,k}$. At first glance our arguments might appear to contradict the aforementioned compression factors 4 and 6 achieved in the case of $G_{q\pm\sqrt{2q}+1} \subsetneq G_{q,4}$ where q is a power of 2, and $G_{q\pm\sqrt{3q}+1} \subsetneq G_{q,6}$ where q is a power of 3. However, we explain why this discrepancy occurs, and how it helps to work in characteristic two and three fields to compress the elements of certain subgroups G_ℓ by a factor $k \approx (k \log q)/\log \ell$, when $\ell \approx q$. In particular, we present torus-based compression methods in characteristic two and three fields that achieve factor-4 and 6 compression, respectively. We should emphasize that previously the only method known to compress by a factor-4 and 6 was to use the trace representation of elements [15, 8]. Our new approach gives us the opportunity to compare the two compression methods and, in fact, has the advantage that computing the decompression functions is essentially free. This yields more efficient exponentiation algorithms compared to the trace-based exponentiation algorithms where decompression is quite costly.

The remainder of this paper is organized as follows. In Section 2, we recall some of the results in the literature and set the notation for the paper. In Section 3, we construct our argument to support the difficulty of obtaining the *optimal* compression factor k for the elements of $G_\ell \subsetneq G_{q,k} \subset \mathbb{F}_{q^k}$, where $\ell \approx q$. We analyze two particular cases in Sections 4 and 5 and show that, in contrast to our pessimistic arguments in Section 3, one can obtain factor-4 and factor-6 compression using torus-based techniques. In Sections 6 and 7 we describe several exponentiation algorithms based on our compression and decompression techniques. In Section 8 we give a comparison of exponentiation algorithms and conclude in Section 9.

2. A review of torus-based compression

Let q be a prime power and $\mathbb{F}_q$ denote the finite field of order q. We denote the trace function $\mathrm{Tr}_{\mathbb{F}_{q^i}/\mathbb{F}_{q^j}} : \mathbb{F}_{q^i} \to \mathbb{F}_{q^j}$ by Tr_{q^i,q^j}.

Let ℓ be a positive integer such that $\gcd(\ell, q) = 1$, and let k be the smallest positive integer such that $q^k \equiv 1 \pmod{\ell}$. Then the order-ℓ group G_ℓ is a subgroup of the cyclotomic subgroup $G_{q,k} \subset \mathbb{F}_{q^k}^*$, where $|G_{q,k}| = \Phi_k(q)$.

Rubin and Silverberg proved that $G_{q,k}$ is isomorphic to the $\mathbb{F}_q$-rational points of an algebraic torus $\mathbb{T}_{q,k}$ of dimension $\varphi(k)$ over $\mathbb{F}_q$. In particular, for $k = 2$ and $k = 6$ they presented explicit compression and decompression algorithms for the elements of $\mathbb{T}_{q,k}(\mathbb{F}_q)$ achieving compression factors $2/\varphi(2) = 2$ and $3 = 6/\varphi(6)$. The compression and decompression maps that correspond

to the case $k = 2$ will be the building blocks in our arguments, so we explicitly state them here for future reference (see [12, 13] for more details).

Let $\mathbb{F}_{q^2} = \mathbb{F}_q[\sigma]/(f(\sigma))$. If q is even, we set $f(\sigma) = \sigma^2 + \sigma + c$ with $c \in \mathbb{F}_q$ and $\mathrm{Tr}_{q^2,2}(c) = 1$. If q is odd, we set $f(\sigma) = \sigma^2 - c$ where $c \in \mathbb{F}_q$ is a quadratic non-residue. Then

$$\begin{aligned} \mathcal{C}: \ G_{q,2} \setminus \{\pm 1\} &\to \mathbb{F}_q \\ g_0 + g_1\sigma &\mapsto \frac{g_0+1}{g_1}, \end{aligned} \tag{2.1}$$

and

$$\begin{aligned} \mathcal{D}: \ \mathbb{F}_q &\to G_{q,2} \\ \alpha &\mapsto \begin{cases} \frac{\alpha+\sigma}{\alpha+1+\sigma} & \text{if } q \text{ is even,} \\ \frac{\alpha+\sigma}{\alpha-\sigma} & \text{if } q \text{ is odd,} \end{cases} \end{aligned} \tag{2.2}$$

define the compression and the decompression maps, respectively [13]. After observing that

$$\begin{aligned} G_{q,2} &= \{g_0 + g_1\sigma : \ g_0, g_1 \in \mathbb{F}_q \text{ and } (g_0+g_1\sigma)^{q+1} = 1\} \\ &= \begin{cases} \{g_0 + g_1\sigma : \ g_0, g_1 \in \mathbb{F}_q \text{ and } g_0^2 + cg_1^2 + g_0g_1 = 1 & \text{if } q \text{ is even;} \\ \{g_0 + g_1\sigma : \ g_0, g_1 \in \mathbb{F}_q \text{ and } g_0^2 - cg_1^2 = 1 & \text{if } q \text{ is odd,} \end{cases} \end{aligned}$$

one can check that $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined, and that

$$\mathcal{D}(\alpha)\mathcal{D}(\beta) = \mathcal{D}\left(\frac{\alpha\beta + c}{\alpha + \beta + 1}\right) \quad \text{if } q \text{ is even,} \tag{2.3}$$

$$\mathcal{D}(\alpha)\mathcal{D}(\beta) = \mathcal{D}\left(\frac{\alpha\beta + c}{\alpha + \beta}\right) \quad \text{if } q \text{ is odd.} \tag{2.4}$$

We note that formulas (2.3) and (2.4) can be used to perform multiplication and exponentiation in $G_{q,2} \setminus \{\pm 1\}$ when working with the compressed representation of elements in $\mathbb{F}_q$.

3. On the (im)possibility of optimal compression

In Section 2 we saw that one can at best hope to compress the elements of $G_{q,k} \subset \mathbb{F}_{q^k}^*$ by a factor $k/\varphi(k)$ which seems to be the optimal compression factor as $|G_{q,k}| \approx q^{\varphi(k)}$. However, it is also known that for $k = 4$ and $k = 6$, one can compress further by a factor of 2 and obtain compression factor $2k/\varphi(k) = k$ for the elements of certain proper subgroups G_ℓ of $G_{q,k}$ [15, 8]. In particular, for $k = 4$ we have $q = 2^m$ and $\ell = q \pm \sqrt{2q} + 1$, and for $k = 6$ we have $q = 3^m$ and $\ell = q \pm \sqrt{3q} + 1$; in both cases, m is odd. Note that, in both cases, $G_\ell \approx q$ and so k is the optimal compression factor. In general, it would be desirable to compress the elements of any order-ℓ subgroup $G_\ell \subsetneq G_{q,k} \subseteq \mathbb{F}_{q^k}^*$ by an *optimal* factor $(k \log q)/\log \ell$ in any characteristic.

We first recall some details on how to achieve compression factor $k = 4$ in characteristic-two fields using the trace representation of elements, and explain why this compression technique does not seem to generalize to fields with characteristic different from two.

Let $q = 2^m, t = \sqrt{2q}$ and $\ell = q \pm t + 1$, where m is odd. We note that $\Phi_4(q) = q^2 + 1 = (q + t - 1)(q - t + 1)$ and $G_\ell \subset G_{q,4} \subset \mathbb{F}_{q^4}^*$ has embedding degree $k = 4$ with respect to q. In [8], it was shown that if $g \in G_\ell$ then the minimal polynomial f_g of g over $\mathbb{F}_q$ is

$$f_g(x) = x^4 - \mathrm{Tr}_{q^4,q}(g)x^3 + \mathrm{Tr}_{q^4,q}(g)^t x^2 - \mathrm{Tr}_{q^4,q}(g)x + 1.$$

One readily deduces that g can be uniquely identified up to conjugation over $\mathbb{F}_q$ from $\mathrm{Tr}_{q^4,q}(g)$, thereby achieving factor-4 compression. A natural extension is to try to represent the elements of G_ℓ with embedding degree 4 using their traces over $\mathbb{F}_q$, where q is not even. We fix parameters (q, ℓ) such that q is a prime power, $\gcd(q, \ell) = 1$, $q^2 + 1 \equiv 0 \pmod{\ell}$, and $q^i \not\equiv 1 \pmod{\ell}$ for

$1 \leq i < 4$. It is shown in [8] that the minimal polynomial f_g of $g \in G_\ell$ over $\mathbb{F}_q$ can be computed as

$$f_g(x) = x^4 - \mathrm{Tr}_{q^4,q}(g)x^3 + (\mathrm{Tr}_{q^4,q}(g^{q+1}) + 2)x^2 - \mathrm{Tr}_{q^4,q}(g)x + 1.$$

Therefore, two $\mathbb{F}_q$-elements (as opposed to only one) are generally required to identify g uniquely up to its conjugates over $\mathbb{F}_q$, unless one of $\mathrm{Tr}_{q^4,q}(g)$ or $\mathrm{Tr}_{q^4,q}(g^{q+1})$ can be obtained from the other. In fact, one can find parameters (q, ℓ), and elements $g_1, g_2 \in G_\ell \subsetneq G_{q,4}$ such that g_1 and g_2 are not conjugates over $\mathbb{F}_q$ but $\mathrm{Tr}_{q^4,q}(g_1) = \mathrm{Tr}_{q^4,q}(g_2)$.

Next, we provide some evidence that, in general, compressing the elements of $G_\ell \subsetneq G_{q,k} \subseteq \mathbb{F}_{q^k}^*$ by an optimal factor $(k \log q)/\log \ell$ might not be possible using *tori-like* techniques. Again, we consider the case $k = 4$ and $\ell \approx q$. Note that $(k \log q)/\log \ell \approx 4$. Let $\mathbb{F}_{q^2} = \mathbb{F}_q[w]/(g(w))$ and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[\sigma]/(f(\sigma))$ for some suitable f and g. Let $g = g_0 + g_1\sigma \in G_\ell$ and recall from Section 2 that if $g \neq \pm 1$ then it can be uniquely identified with an element $\alpha \in \mathbb{F}_{q^2}$ if $f(\sigma)$ is of the form $\sigma^2 - c$ or $\sigma^2 + \sigma + c$; see (2.1) and (2.2). More precisely, if $g = g_0 + g_1\sigma$ then $\alpha = (g_0+1)/g_1$ and $g = \mathcal{D}(\alpha)$, where $\mathcal{D}(\alpha) = (\alpha+\sigma)/(\alpha-\sigma)$ if q is even, and $\mathcal{D}(\alpha) = (\alpha+\sigma)/(\alpha+1+\sigma)$ if q is odd. Let $\alpha = a + bw$ for some $a, b \in \mathbb{F}_q$. Note that the compression of g into (a, b) does not utilize the fact that g lies in a proper subgroup G_ℓ of $G_{q,4}$. Therefore, one might try to compress g further into b (or a), by using the relation $g^\ell = 1$ to obtain an expression for one of a and b in terms of the other. For example, the most naive way would be to use the relation $g^\ell = 1$ and obtain a polynomial $P(x, y) \in \mathbb{F}_q[x, y]$ such that $P(a, b) = 0$. Then we would hope to find a among the roots of $P(x, b) = 0$. Since we also want the corresponding decompression function to be efficiently computable and *almost* one-to-one we might ask the following question.

Question: What is the minimum expected degree and sparsity of a polynomial $P(x, y) \in \mathbb{F}_q[x, y]$ such that (i) for (*almost*) all $b \in \mathbb{F}_q$ there exists an (*almost*) unique solution $a \in \mathbb{F}_q$ to $P(x, b) = 0$; and (ii) for $\alpha = a + bw$ we have $\mathcal{D}(\alpha) \in G_\ell$.

Remark 3.1. Given $\binom{n+m}{n}$ pairs $(X_i, z_i) \in \mathbb{F}_q^m \times \mathbb{F}_q$, an nth degree polynomial P in m variables can be constructed such that $P(X_i) = z_i$. Note that $\binom{n+m}{n}$ is the number of ways of choosing n elements from a set of $m+1$ elements with repetitions allowed, which is therefore the maximum number of monomials in P. Hence, when $m = 2$ we would expect $\deg(P(x, y)) \leq \sqrt{\ell} \approx \sqrt{q}$. We would even expect, in general, that $\deg(P(x, y)) \approx \sqrt{q}$ unless the relation $g^\ell = 1$ can be manipulated towards obtaining a polynomial $P(x, y)$ of a rather special form. This shows that tori-like techniques described above will likely fail to produce efficient compression and decompression functions.

4. Factor-4 compression in characteristic two

Let $q = 2^m$, m odd, $t = \sqrt{2q}$, $\ell = q + 1 - t$ and $\bar{\ell} = q + 1 + t$. Then

$$\begin{aligned} q^4 - 1 &= (q^2 - 1)(q^2 + 1) \\ &= (q^2 - 1)(q + 1 - t)(q + 1 + t). \end{aligned}$$

Let $G_\ell \subset G_{q,4} \subset \mathbb{F}_{q^4}^*$ and $G_{\bar{\ell}} \subset G_{q,4} \subset \mathbb{F}_{q^4}^*$ be subgroups such that $|G_{q,4}| = q^2 + 1$, $|G_\ell| = \ell$ and $|G_{\bar{\ell}}| = \bar{\ell}$. In this section, we set $\mathbb{F}_{q^2} = \mathbb{F}_q[w]/(w^2 + w + c_0)$ and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[\sigma]/(\sigma^2 + \sigma + c_1)$. We must have $\mathrm{Tr}_{q,2}(c_0) = \mathrm{Tr}_{q^2,2}(c_1) = 1$.

Lemma 4.1. *Let* $\mathbb{F}_{q^2} = \mathbb{F}_q[w]/(w^2 + w + c_0)$ *and* $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[\sigma]/(\sigma^2 + \sigma + c_1)$ *with* $\mathrm{Tr}_{q,2}(c_0) = \mathrm{Tr}_{q^2,2}(c_1) = 1$. *Then*

$$\sigma^q + \sigma = \sum_{i=0}^{m-1} c_1^{2^i} = u_0 + u_1 w, \tag{4.1}$$

$$\sigma^t + \sigma = \sum_{i=0}^{(m-1)/2} c_1^{2^i} = u_2 + u_3 w, \tag{4.2}$$

$$w^q + w = 1, \tag{4.3}$$

$$w^t + w = \sum_{i=0}^{(m-1)/2} c_0^{2^i} = u_4, \tag{4.4}$$

for some $u_i \in \mathbb{F}_q$. *In particular,* $u_1 = 1$.

Proof. The equalities can be proven by repeatedly squaring the equations $\sigma^2 + \sigma = c_1$ and $w^2 + w = c_0$. We have $u_1 = 1$ since

$$1 = \mathrm{Tr}_{q^2,2}(c_1) = (\sigma + \sigma^q) + (\sigma + \sigma^q)^q = u_1(w + w^q) = u_1.$$

□

We furthermore assume throughout this section that $\sigma^2 + \sigma = c_1 = u_5 + u_6 w$, where $u_5, u_6 \in \mathbb{F}_q$.

Let $g = g_0 + g_1\sigma \in G_{q,4}$. We already know from Section 2 that if $g \neq 1$ then g can be compressed to an element $\alpha = (g_0 + 1)/g_1 \in \mathbb{F}_{q^2}$, and that a compressed element $\alpha \in \mathbb{F}_{q^2}$ can be decompressed to obtain $g = (\alpha + \sigma)/(\alpha + 1 + \sigma) \in G_{q,4} \setminus \{1\}$. Our objective is to show that $g \in \{G_\ell, G_{\bar{\ell}}\} \setminus \{1\}$ can further be compressed to $b \in \mathbb{F}_q$, and that a compressed $b \in \mathbb{F}_q$ can be decompressed to obtain $g \in \{G_\ell, G_{\bar{\ell}}\} \setminus \{1\}$. The following theorem plays a key role.

Theorem 4.2. *Let* $g = (\alpha + \sigma)/(\alpha + 1 + \sigma) \in G_{q,4} \setminus \{1\}$ *where* $\alpha = a + bw \in \mathbb{F}_{q^2}$ *for some* $a, b \in \mathbb{F}_q$. *If* $g \in G_\ell$ *then* a *is a root of the polynomial*

$$P_1(x, b) = x^t + x + b^{t+1} + (u_0 + u_4)b^t + (u_0 + u_3 + 1)b + (u_0 u_3 + u_2 + u_6),$$

where the u_i*'s are as specified in Lemma 4.1. If* $g \in G_{\bar{\ell}}$ *then* a *is a root of the polynomial*

$$P_1(x, b) = x^t + x + b^{t+1} + (u_0 + u_4)b^t + (u_0 + u_3 + 1)b + (u_0 u_3 + u_2 + u_6 + 1).$$

Proof. Let $g = (\alpha + \sigma)/(\alpha + 1 + \sigma) \in G_\ell \setminus \{1\}$. After expanding and simplifying $g^{q+1-t} = 1$, we find that $\alpha + \sigma$ is a root of

$$P(x) = x^{q+t} + x^{q+1} + x^{t+1} + x^t.$$

Now, writing $\alpha = a + bw$ for some $a, b \in \mathbb{F}_q$ and simplifying $P(\alpha + \sigma) = 0$ gives us

$$\begin{aligned} P(\alpha) &= a^2 + ab + (\sigma^q + \sigma)a + (w^2 + w)b^2 + ba^t + w^t b^{t+1} \\ &\quad + (w(\sigma^q + \sigma) + \sigma^t + \sigma)b + (\sigma^q + \sigma + 1)a^t \\ &\quad + (w^t(\sigma^q + \sigma + 1))b^t + (\sigma^q(\sigma^t + \sigma) + \sigma^t(\sigma + 1)) \\ &= a^2 + ab + (u_0 + w)a + c_0 b^2 + ba^t + (u_4 + w)b^{t+1} \\ &\quad + ((u_0 + u_3 + 1)w + (c_0 + u_2))b + (u_0 + 1 + w)a^t \\ &\quad + ((u_0 + u_4)w + (c_0 + u_0 u_4 + u_4))b^t + (u_0 u_3 + u_2 + u_6)w \\ &\quad + (c_0 u_3 + u_0 u_2 + u_2 + u_5) \\ &= P_0(a, b) + P_1(a, b)w = 0, \end{aligned}$$

where

$$\begin{aligned} P_0(a, b) &= a^t b + (u_0 + 1)a^t + a^2 + ab + u_0 a + u_4 b^{t+1} \\ &\quad + (c_0 + u_0 u_4 + u_4)b^t + c_0 b^2 + (c_0 + u_2)b \\ &\quad + (c_0 u_3 + u_0 u_2 + u_2 + u_5), \\ P_1(a, b) &= a^t + a + b^{t+1} + (u_0 + u_4)b^t + (u_0 + u_3 + 1)b \\ &\quad + (u_0 u_3 + u_2 + u_6). \end{aligned}$$

Hence, if $g = \frac{\alpha+\sigma}{\alpha+1+\sigma} \in G_\ell$ for some $\alpha = a + bw \in \mathbb{F}_{q^2}$ with $a, b \in \mathbb{F}_q$, we must have $P_0(a,b) = P_1(a,b) = 0$. In particular, a must be a root of the polynomial

$$P_1(x) = P_1(x,b) \;=\; x^t + x + b^{t+1} + (u_0+u_4)b^t + (u_0+u_3+1)b + (u_0u_3+u_2+u_6).$$

The case when $g \in G_{\overline{\ell}} \setminus \{1\}$ can be proved similarly. □

Lemma 4.3. *Let $P_1(x) = x^t + x + u \in \mathbb{F}_q[x]$. Then $P_1(x) = 0$ has a solution in $\mathbb{F}_q$ if and only if $\mathrm{Tr}_{q,2}(u) = 0$. If $\mathrm{Tr}_{q,2}(u) = 0$ then $P_1(x) = 0$ has exactly two solutions a_0, a_1 in $\mathbb{F}_q$, and $a_1 = a_0 + 1$.*

Proof. We first prove that $P_1(x) = 0$ has a solution in $\mathbb{F}_q$ if and only if $\mathrm{Tr}_{q,2}(u) = 0$. Suppose that $P_1(x) = x^t + x + u = 0$ has a solution, say $a \in \mathbb{F}_q$. Then

$$\mathrm{Tr}_{q,2}(u) \;=\; \mathrm{Tr}_{q,2}(a^t + a) = \mathrm{Tr}_{q,2}(a)^t + \mathrm{Tr}_{q,2}(a) = 0. \tag{4.5}$$

Now, define a *half-trace* function $H:\ \mathbb{F}_q \to \mathbb{F}_q$ as follows[1]

$$H(u) = \sum_{i=0}^{(m-1)/2} u^{2^i}. \tag{4.6}$$

Then $H(u)^t + H(u) = u + \mathrm{Tr}_{q,2}(u)$, and so $H(u) \in \mathbb{F}_q$ is a solution to $P_1(x) = 0$ when $\mathrm{Tr}_{q,2}(u) = 0$.

Next we prove that if $\mathrm{Tr}_{q,2}(u) = 0$ then $P_1(x) = 0$ has exactly two solutions, namely $H(u)$ and $H(u)+1$. We first consider the case $m = 4i+3$. Note that $q = 2^m$ and $t = \sqrt{2q} = 2^{2i+2}$. Let us fix a normal basis to represent $\mathbb{F}_q$ as an m-dimensional vector space over $\mathbb{F}_2$. In this representation, we may set

$$\begin{aligned} x &= (x_0, x_1, \ldots, x_{2i}, x_{2i+1}, x_{2i+2}, x_{2i+3}, \ldots, x_{4i+1}, x_{4i+2}),\\ x^t &= (x_{2i+1}, x_{2i+2}, \ldots, x_{4i+1}, x_{4i+2}, x_0, x_1, \ldots, x_{2i-1}, x_{2i}),\\ u &= (u_0, u_1, \ldots, u_{2i}, u_{2i+1}, u_{2i+2}, u_{2i+3}, \ldots, u_{4i+1}, u_{4i+2}). \end{aligned}$$

Then $P_1(x) = 0$ has a solution if and only if the linear system of equations determined by

$$x_j + x_{2i+2+j} \;=\; u_{2i+2+j},\ 0 \le j \le 2i, \tag{4.7}$$

$$x_{2i+1+j} + x_j \;=\; u_j,\ 1 \le j \le 2i+1, \tag{4.8}$$

$$x_{2i+2} + x_0 \;=\; u_0 \tag{4.9}$$

has a solution $X = (x_0, x_1, \ldots, x_{4i+2}) \in \mathbb{F}_2^m$. We can see from (4.7) and (4.8) that a choice of $x_0 \in \{0,1\}$ fixes x_j for all $1 \le j \le 4i+2$, and hence fixes two vectors X_0 and X_1 in $\mathbb{F}_2^m$. Now, it follows from (4.9) that $P_1(x) = 0$ has a solution if and only if $x_0 + x_{2i+1} = u_0$. Therefore, $P_1(x) = 0$ has at most two solutions in $\mathbb{F}_q$. In particular, when $P_1(x) = 0$ has a solution a_0 then there are exactly two solutions and the other solution is $a_1 = a_0 + 1$ since

$$P_1(a_0+1) = (a_0+1)^t + (a_0+1) + u = a_0^t + a_0 + u = 0.$$

The case $m = 4i+1$ can be similarly proven. □

Now, we are ready to describe our compression/decompression maps that achieve factor-$(4\log q/(1+\log q))$ compression.

Theorem 4.4. *For some fixed representation of $\mathbb{F}_q$ as an m-dimensional vector space over $\mathbb{F}_2$, let j be a coordinate position such that the vector representations of β and $\beta+1$ differ in the jth coordinate position for all $\beta \in \mathbb{F}_q$. Let $G \in \{G_\ell, G_{\overline{\ell}}\}$. Define a compression map*

$$\begin{aligned} \mathcal{C} : G \setminus \{1\} &\to \{0,1\} \times \mathbb{F}_q\\ g &\mapsto (i, b), \end{aligned} \tag{4.10}$$

[1]The definition is similar to the one in [6, Section 3.6.2]

where $g = g_0 + g_1\sigma$, $(g_0+1)/g_1 = a + bw$, and i is the jth bit in the vector representation of a. And define a decompression map

$$\begin{aligned} \mathcal{D} : \{0,1\} \times \mathbb{F}_q &\to G \setminus \{1\} \\ (i,b) &\mapsto (\alpha+\sigma)/(\alpha+1+\sigma), \end{aligned} \tag{4.11}$$

where $\alpha = a + bw$, and a is one of the two roots of $P_1(x,b)$ (see Theorem 4.2) whose jth bit when represented as a vector over $\mathbb{F}_2$ is equal to i. Then $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined. Moreover, if $\mathcal{D}(0,b) \in G$ then $\mathcal{D}(1,b) \in G$ and $\mathcal{D}(0,b)\mathcal{D}(1,b) = 1$.

Proof. It follows from Theorem 4.2 and Lemma 4.3 that $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined. Now, by Lemma 4.3, $P_1(x,b)$ has exactly 2 solutions a_0 and a_1 in $\mathbb{F}_q$, and $a_1 = a_0 + 1$. Note that since $g = \frac{\alpha+\sigma}{\alpha+1+\sigma} \in G$ with $\alpha = a + bw$ corresponding to (a_0, b), the element $h = \frac{\alpha+1+\sigma}{\alpha+\sigma}$ corresponds to (a_1, b) and is in fact the multiplicative inverse of g. It follows that $\mathcal{D}(0,b)\mathcal{D}(1,b) = 1$. □

Remark 4.5. The polynomials $P_0(x,y)$ and $P_1(x,y)$ are both of degree $(t+1) \approx \sqrt{q}$ which is in accordance with Remark 3.1. However, $P_1(x,b)$ is very sparse and moreover it is easy to find a root $a \in \mathbb{F}_q$, in contrast to what one would expect in general for high-degree polynomials.

5. Factor-6 compression in characteristic three

Let $q = 3^m$, $m \equiv 5 \pmod{12}$, $t = \sqrt{3q}$ and $\ell = q + 1 - t$. Then

$$\begin{aligned} q^6 - 1 &= (q^3-1)(q^3+1) \\ &= (q^3-1)(q+1)(q^2-q+1) \\ &= (q^3-1)(q+1)(q+1-t)(q+1+t). \end{aligned}$$

Let $G_\ell \subset G_{q,6} \subset \mathbb{F}_{q^6}^*$ be subgroups such that $|G_{q,6}| = q^2 - q + 1$ and $|G_\ell| = \ell$. Since $f(w) = w^3 - w - 1$ has splitting field $\mathbb{F}_{3^3}$ and $\gcd(3,m) = 1$, f is irreducible over $\mathbb{F}_q$ and we set $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3 - w - 1)$. We also let $c_0 \in \mathbb{F}_{q^3}$ be a quadratic non-residue and set $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 - c_0)$.

Lemma 5.1. *Let $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3 - w - 1)$ and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 - c_0)$ where $c_0 \in \mathbb{F}_{q^3}$ is a quadratic non-residue. Then*

$$\begin{aligned} \sigma^t &= c_1\sigma, \\ \sigma^q &= c_2\sigma, \\ \sigma^{q^2} &= c_3\sigma, \\ w^t &= w, \\ w^{2t} &= w^2, \\ w^q &= w + 2, \\ w^{2q} &= w^2 + w + 1, \\ w^{q^2} &= w + 1, \end{aligned}$$

for some $c_1, c_2, c_3 \in \mathbb{F}_{q^3}$.

Proof. The equalities can be proven by using the defining equations of σ and w, and noting that $w^{3^{2k}} = w + 2k$ and $w^{3^{2k+1}} = w + 1 + 2k$. □

We furthermore assume throughout this section that $c_i = u_{3i}+u_{3i+1}w+u_{3i+2}w^2$ for $i = 0, 1, 2, 3$, where $u_j \in \mathbb{F}_q$.

Let $g = g_0 + g_1\sigma \in G_{q,6}$. We already know from Section 2 that if $g \neq \pm 1$ then g can be compressed to an element $\alpha = (g_0 + 1)/g_1 \in \mathbb{F}_{q^3}$, and that a compressed $\alpha \in \mathbb{F}_{q^3}$ can be decompressed to obtain $g = (\alpha + \sigma)/(\alpha - \sigma) \in G_{q,6} \setminus \{\pm 1\}$. Our objective is to show that $g \in G_\ell$ can be compressed to $c \in \mathbb{F}_q$, and that a compressed $c \in \mathbb{F}_q$ can be decompressed to obtain $g \in G_\ell \setminus \{\pm 1\}$. The following theorem plays a key role.

Theorem 5.2. *Let $g = (\alpha+\sigma)/(\alpha-\sigma) \in G_\ell\setminus\{\pm 1\}$ where $\alpha = a+bw+cw^2$ for some $a, b, c \in \mathbb{F}_q$. Then $(x_1, x_2, x_3, x_4, x_5, x_6) = (a, a^t, b, b^t, c, c^t)$ is a root of each f_i and g_i for $i = 0, 1, 2$, where*

$$\begin{aligned}
f_0 \;=\;& u_0u_3u_6 + u_0u_4u_8 + u_0u_5u_7 + u_1u_3u_8 + u_1u_4u_7 + u_1u_5u_6 \\
&+u_1u_5u_8 + u_2u_3u_7 + u_2u_4u_6 + u_2u_4u_8 + u_2u_5u_7 + u_2u_5u_8 \\
&+u_3x_1^2 + (u_4 + 2u_5)x_3^2 + (u_3 + 2u_4 + 2u_5)x_5^2 + (2u_6 + 2)x_1x_2 \\
&+(2u_7 + 2u_8 + 2)x_5x_6 + (2u_6 + 2u_8 + 2)x_3x_6 + (2u_8 + 1)x_2x_3 \\
&+(2u_7 + 2)x_2x_5 + (2u_6 + 2u_8 + 2)x_4x_5 + (2u_3 + 2u_5)x_1x_3 \\
&+(u_3 + 2u_4 + u_5)x_1x_5 + 2u_3x_3x_5 + 2u_7x_3x_4 + 2u_7x_1x_6 \\
&+2u_8x_1x_4,
\end{aligned}$$

$$\begin{aligned}
f_1 \;=\;& u_0u_3u_7 + u_0u_4u_6 + u_0u_4u_8 + u_0u_5u_7 + u_0u_5u_8 + u_1u_3u_6 \\
&+u_1u_3u_8 + u_1u_4u_7 + u_1u_4u_8 + u_1u_5u_6 + u_1u_5u_7 + u_1u_5u_8 \\
&+u_2u_3u_7 + u_2u_3u_8 + u_2u_4u_6 + u_2u_4u_7 + u_2u_4u_8 + u_2u_5u_6 \\
&+u_2u_5u_7 + 2u_2u_5u_8 + u_4x_1^2 + (2u_3 + u_4)x_3^2 + (2u_3 + u_5)x_5^2 \\
&+(2u_6 + 2u_7 + u_8 + 1)x_5x_6 + (2u_7 + 2u_8 + 1)x_3x_4 \\
&+(2u_6 + 2u_7 + 2u_8 + 2)x_3x_6 + (2u_6 + 2u_8 + 2)x_2x_3 \\
&+(2u_7 + 2u_8 + 2)x_2x_5 + (2u_6 + 2u_7 + 2u_8 + 1)x_4x_5 \\
&+(2u_7 + 2u_8)x_1x_6 + (2u_6 + 2u_8 + 2)x_1x_4 + u_3x_1x_5 \\
&+(2u_3 + 2u_4 + 2u_5)x_1x_3 + 2u_4x_3x_5 + 2u_7x_1x_2,
\end{aligned}$$

$$\begin{aligned}
f_2 \;=\;& u_0u_3u_8 + u_0u_4u_7 + u_0u_5u_6 + u_0u_5u_8 + u_1u_3u_7 + u_1u_4u_6 \\
&+u_1u_4u_8 + u_1u_5u_7 + u_1u_5u_8 + u_2u_3u_6 + u_2u_3u_8 + u_2u_4u_7 \\
&+u_2u_4u_8 + u_2u_5u_6 + u_2u_5u_7 + u_2u_5u_8 + 2u_7x_2x_3 + u_5x_1^2 \\
&+(u_3 + 2u_4 + u_5)x_3^2 + (2u_3 + 2u_4)x_5^2 + (2u_4 + 2u_5)x_1x_3 \\
&+(2u_3 + u_4)x_1x_5 + 2u_8x_1x_2 + 2u_5x_3x_5 + 2u_7x_1x_4 \\
&+(2u_6 + 2u_7 + 2u_8 + 1)x_5x_6 + (2u_6 + 2u_8 + 2)x_3x_4 \\
&+(2u_7 + 2u_8 + 1)x_3x_6 + (2u_6 + 2u_8 + 2)x_2x_5 \\
&+(2u_7 + 2u_8 + 2)x_4x_5 + (2u_6 + 2u_8 + 2)x_1x_6,
\end{aligned}$$

$$\begin{aligned}
g_0 \;=\;& u_2u_7u_{11} + u_2u_8u_{10} + u_2u_8u_{11} + u_0u_6u_9 + u_1u_7u_{10} + u_0u_7u_{11} \\
&+u_0u_8u_{10} + u_1u_6u_{11} + u_2u_7u_9 + u_1u_8u_9 + u_1u_8u_{11} + u_2u_6u_{10} \\
&+(2u_9 + 2 + u_6)x_1^2 + (u_9 + u_6 + 2u_8 + u_{11})x_3x_1 \\
&+(2u_9 + 1 + 2u_8 + u_{10} + 2u_{11} + 2u_7 + u_6)x_1x_5 \\
&+(u_7 + u_8 + 2u_{10} + u_{11} + 1)x_3^2
\end{aligned}$$

$$+(2u_6+2u_7+2u_9+u_{10}+u_{11}+2)x_5^2+(2u_6+u_9+1)x_3x_5$$

$$\begin{aligned}
g_1 &= u_0u_6u_{10}+u_0u_7u_9+u_0u_7u_{11}+u_0u_8u_{10}+u_0u_8u_{11}+u_1u_6u_9\\
&\quad +u_1u_6u_{11}+u_1u_7u_{10}+u_1u_7u_{11}+u_1u_8u_9+u_1u_8u_{10}+u_1u_8u_{11}\\
&\quad +u_2u_6u_{10}+u_2u_6u_{11}+u_2u_7u_9+u_2u_7u_{10}+u_2u_7u_{11}+u_2u_8u_9\\
&\quad +u_2u_8u_{10}+2u_2u_8u_{11}+(u_7+2u_{10})x_1^2\\
&\quad +(u_6+u_7+2u_8+u_9+2u_{10})x_3^2\\
&\quad +(u_7+2u_8+u_9+2u_{11}+2)x_5^2\\
&\quad +(2u_6+u_7+2u_8+u_9+u_{10}+u_{11}+1)x_1x_3\\
&\quad +(2u_6+u_8+2u_9)x_1x_5+(2u_7+u_{10})x_3x_5,
\end{aligned}$$

$$\begin{aligned}
g_2 &= u_1u_6u_{10}+u_1u_7u_9+u_1u_7u_{11}+u_1u_8u_{10}+u_1u_8u_{11}+u_2u_6u_9\\
&\quad +u_2u_6u_{11}+u_2u_7u_{10}+u_2u_7u_{11}+u_0u_6u_{11}+u_0u_7u_{10}+u_0u_8u_9\\
&\quad +u_0u_8u_{11}+u_2u_8u_9+u_2u_8u_{10}+u_2u_8u_{11}+(u_8+2u_{11})x_1^2\\
&\quad +(2u_7+u_8+u_{10}+u_{11})x_1x_3\\
&\quad +(2u_6+2u_7+u_9+2u_{10}+1)x_1x_5\\
&\quad +(u_6+u_7+u_8+2u_9+u_{10}+2u_{11}+2)x_3^2\\
&\quad +(2u_6+u_8+u_9+u_{10}+1)x_5^2+(2u_8+u_{11})x_3x_5,
\end{aligned}$$

and where $c_i = u_{3i}+u_{3i+1}w+u_{3i+2}w^2$*, for* $i=0,1,2,3$*, are as specified in Lemma 5.1.*

Proof. Let $g=(\alpha+\sigma)/(\alpha-\sigma)\in G_\ell\setminus\{\pm1\}$. Expanding the equations $g^{q^2-q+1}=1$ and $g^{q+1-t}=1$ and simplifying using Lemma 5.1 yields the polynomials $f_i, g_i\in\mathbb{F}_q[x_1,x_2,\ldots,x_6]$ for $i=0,1,2$ such that $(x_1,x_2,x_3,x_4,x_5,x_6)=(a,a^t,b,b^t,c,c^t)$ is a root of each f_i and g_i, as required. □

Theorem 5.2 suggests that one can compress an element $g\in G_\ell\setminus\{\pm1\}$ to an element $c\in\mathbb{F}_q$. Given a compressed representation c of an element g, one might reconstruct g by finding a common root (a,a^t,b,b^t,c,c^t) of the f_i and g_i. This may be achieved by constructing a Groebner basis of the ideal in $\mathbb{F}_q[x_1,x_2,\ldots,x_6]$ generated by f_i and g_i evaluated at $x_5=c$, $x_6=c^t$ for $i=0,1,2$. The next corollary shows that this is indeed possible in the case that $c_0=-1$.

Corollary 5.3. *Let* $\mathbb{F}_{q^3}=\mathbb{F}_q[w]/(w^3-w-1)$ *and* $\mathbb{F}_{q^6}=\mathbb{F}_{q^3}[\sigma]/(\sigma^2+1)$*. Let* f_i, g_i *be as in Theorem 5.2. Then a Groebner basis of the ideal* $\langle f_0,f_1,f_2,g_0,g_1,g_2\rangle$ *in* $\mathbb{F}_q[x_1,x_2,\ldots,x_6]$ *is*

$$\begin{aligned}
P_1 &= x_1+2x_3^2x_4+2x_3x_4x_5+x_3+x_5^2x_6+2x_5,\\
P_2 &= x_2+2x_3^3x_4^2+2x_4^2x_5^3+x_4x_5^3x_6+x_5^3x_6^2+2x_6,\\
P_3 &= x_3^2x_4x_5+2x_3^2+x_3x_4x_5^2+2x_3x_5+2x_5^3x_6+2x_5^2+2,\\
P_4 &= x_3x_4^2x_5^2+x_3x_4x_5+x_3+2x_4^2x_5^3+2x_4x_5^3x_6+2x_4x_5^2\\
&\quad +2x_5^3x_6^2+2x_5+2x_6,\\
P_5 &= x_3x_6+2x_4x_5+2x_5x_6+1,\\
P_6 &= x_4^3x_5^3+2x_4x_5^3x_6^2+2x_5^3x_6^3+2x_6^2+2.
\end{aligned}$$

Proof. If one sets $c_0=-1$ in Theorem 5.2 then $g_1=g_2=0$, and the polynomials f_i and g_0 simplify to

$$f_0 = 2x_1^2+x_1x_3+2x_1x_5+x_2x_3+2x_2x_5+x_3x_5+2x_5^2+2x_5x_6+2,$$

$$\begin{aligned} f_1 &= x_1x_3 + 2x_1x_5 + 2x_2x_5 + x_3^2 + x_3x_4 + 2x_4x_5 + x_5^2 + 2x_5x_6, \\ f_2 &= x_1x_5 + 2x_3^2 + x_3x_6 + 2x_4x_5 + x_5^2 + 2x_5x_6, \\ g_0 &= 2x_1x_5 + x_3^2 + 2x_5^2 + 1. \end{aligned}$$

It can be verified using Magma with the commands

```
R < x1, x2, x3, x4, x5, x6 >:= PolynomialRing(FiniteField(3), 6);
B := [R!f0, R!f1, R!f2, R!g0];
I := ideal < R|B >;
GroebnerBasis(I);
```

that a Groebner basis of the ideal $\langle f_0, f_1, f_2, g_0\rangle$ in $\mathbb{F}_q[x_1, x_2, \ldots, x_6]$ is determined by the P_i's, as required. □

5.1. **Decompression procedure.** Let $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3 - w - 1)$ and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 + 1)$. Let $g = (\alpha + \sigma)/(\alpha - \sigma) \in G_\ell \setminus \{\pm 1\}$ where $\alpha = a + bw + cw^2$ for some $a, b, c \in \mathbb{F}_q$. By Theorem 5.2 and Corollary 5.3, b^t must be a root of

$$P_6(x_4) = c^3x_4^3 + 2c^{2t+3}x_4 + 2(c^{3(t+1)} + c^{2t} + 1). \tag{5.1}$$

In fact, there are exactly three roots of $P_6(x_4)$ in $\mathbb{F}_q$, and if r is a root then the other two roots are given by $r \pm c^t$. Therefore, if c is given, b^t can be determined uniquely up to 3 elements, that is, $b^t \in \{r, r - c^t, r + c^t\}$. Once b^t is fixed, one can solve for b uniquely by using $P_5(x_3) = 0$, where $P_5(x_3)$ is obtained by evaluating P_5 at $x_4 = b^t$, $x_5 = c$, $x_6 = c^t$ (see Corollary 5.3), or by using the fact that $b \mapsto b^t$ is a Frobenius map. Having determined b, b^t, c and c^t we can use $P_2(x_2) = 0$, where $P_2(x_2)$ is obtained by evaluating P_2 at $x_3 = b$, $x_4 = b^t$, $x_5 = c$, $x_6 = c^t$ (see Corollary 5.3), to solve for a^t uniquely. Finally, a can be determined either by using $P_1(x_1) = 0$, where $P_1(x_1)$ is obtained by evaluating P_1 at $x_3 = b$, $x_4 = b^t$, $x_5 = c$, $x_6 = c^t$ (see Corollary 5.3), or by using the fact that $a \mapsto a^t$ is a Frobenius map.

To summarize, suppose that $g \in G_\ell \setminus \{\pm 1\}$ and $g = (\alpha + \sigma)/(\alpha - \sigma)$ with $\alpha = a + bw + cw^2$. If c is given, then the three pairs (x_{1h}, x_{3h}), $h = 1, 2, 3$ can be efficiently determined such that $(a, b, c) \in \{(x_{1h}, x_{3h}, c) : \ h = 1, 2, 3\}$. In fact, one can check that $c \neq 0$ and

$$\{(x_{1h}, x_{3h}, c) : \ h = 1, 2, 3\} = \{(a, b, c), (a - b + c, b + c, c), (a + b + c, b - c, c)\}.$$

Suppose now that we have fixed some representation of $\mathbb{F}_q$ as an m-dimensional vector space over $\mathbb{F}_3$. Then there must exist a *smallest index* j such that exactly one of x_{3h}'s j'th trit is equal to b's j'th trit, say $i \in \{0, 1, 2\}$, when they are represented as vectors over $\mathbb{F}_3$. This yields one-to-one compression/decompression maps that achieve factor-$(6 \log q)/(2 + \log q)$ compression.

Theorem 5.4. *Define a compression map*

$$\begin{aligned} \mathcal{C} : G_\ell \setminus \{\pm 1\} &\to \{0, 1, 2\} \times \mathbb{F}_q \\ g &\mapsto (i, c), \end{aligned} \tag{5.2}$$

where $g = g_0 + g_1\sigma$, $(g_0 + 1)/g_1 = a + bw + cw^2$, *and* i *is defined above. Define a decompression map*

$$\begin{aligned} \mathcal{D} : \{0, 1, 2\} \times \mathbb{F}_q &\to G_\ell \setminus \{\pm 1\} \\ (i, c) &\mapsto (\alpha + \sigma)/(\alpha - \sigma), \end{aligned} \tag{5.3}$$

where $\alpha = a + bw + cw^2$, *and* a, b *can be constructed as described above. Then* $\mathcal{C}$ *and* $\mathcal{D}$ *are inverses of each other when they are defined. Moreover, if* $\mathcal{D}(0, c) \in G_\ell \setminus \{\pm 1\}$ *then* $\mathcal{D}(i, b) \in G_\ell \setminus \{\pm 1\}$ *for* $i = 1, 2$, *and* $\mathcal{D}(0, c)\mathcal{D}(1, c)\mathcal{D}(2, c) = 1$.

Proof. It is clear from our arguments above that $\mathcal{C}$ and $\mathcal{D}$ are inverses of each other when they are defined. Now, let $c \in \mathbb{F}_q^*$ be such that $g = (\alpha+\sigma)/(\alpha-\sigma) \in G_\ell \setminus \{\pm 1\}$, where $\alpha = a+bw+cw^2$. Let i_1 be the jth trit of b, where j is the smallest index such that if $(b+c)$'s j'th trit is i_2 and $(b-c)$'s j'th trit is i_3, then i_1, i_2, i_3 are pairwise different. It follows from our arguments above that the decompression function satisfies $\mathcal{D}(i_h, c) = g_h$, where $g_h = (\alpha_h - \sigma)/(\alpha_h + \sigma)$, and $\alpha_1 = a + bw + cw^2, \alpha_2 = (a-b+c) + (b+c)w + cw^2, \alpha_3 = (a+b+c) + (b-c)w + cw^2$. Moreover, one can check that $g_1 = g, g_2 = g^{-q} = g^{q^4}$, and $g_3 = g^{q^2}$, that is $g_1 g_2 g_3 = 1$, as required. □

Remark 5.5. It would be interesting to prove similar results for $q = 3^m$ where $m \not\equiv 5 \pmod{12}$, and for any quadratic non-residue $c_0 \in \mathbb{F}_{q^3}$, $c_0 \neq -1$. The main difficulty when $c_0 \neq -1$ seems to be that the polynomials f_i, g_i are defined strictly over $\mathbb{F}_q$ rather than over $\mathbb{F}_3$ which is the case when $c_0 = -1$.

6. Factor-4 compression and exponentiation algorithms

In this section, we analyze the efficiency of the compression and decompression methods proposed in Section 4. The efficiency of these methods matters because given a compressed representation of an element, one can consider a variety of exponentiation algorithms that can work directly with that compressed representation, or with partially or fully decompressed representations of the element.

We first show that compression and decompression can be achieved at a negligible cost. Then we describe two exponentiation algorithms and provide a performance comparison.

6.1. Compression/decompression costs.

Let $q = 2^m$, m odd, $t = \sqrt{2q}$ and $\ell = q + 1 - t$. Let $G_\ell \subset \mathbb{F}_{q^4}^*$ be the subgroup with $|G_\ell| = \ell$. Let $\mathbb{F}_{q^2} = \mathbb{F}_q[w]/(w^2 + w + c_0)$ and $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[\sigma]/(\sigma^2+\sigma+c_1)$, where $\mathrm{Tr}_{q,2}(c_0) = \mathrm{Tr}_{q^2,2}(c_1) = 1$. We further assume that $\mathbb{F}_q$ is represented as an m-dimensional vector space over $\mathbb{F}_2$ via a polynomial basis $\{1, z, \ldots, z^{m-1}\}$.

We first show that the compression and decompression maps described in Theorem 4.4 are very efficiently computable.

Lemma 6.1. *Let $P_1(x) = x^t + x + u \in \mathbb{F}_q[x]$. If $P_1(x) = 0$ has a solution in $\mathbb{F}_q$ then it can be computed at a cost of $(m-1)/2$ squarings and $(m-1)/2$ additions in $\mathbb{F}_q$. If storage for m $\mathbb{F}_q$-elements is available then finding the $\mathbb{F}_q$-solutions of $P_1(x) = 0$ in $\mathbb{F}_q$ requires on average $m/2$ additions in $\mathbb{F}_q$.*

Proof. Let $u = \sum_{i=0}^{m-1} u_i z^i$ and suppose that $P_1(x) = x^t + x + u = 0$ has a solution in $\mathbb{F}_q$. It follows from Lemma 4.3 that the solutions in $\mathbb{F}_q$ are given by $H(u)$ and $H(u)+1$, where $H(u) = \sum_{i=0}^{(m-1)/2} u^{2^i}$, which can be computed at a cost of $(m-1)/2$ squarings and $(m-1)/2$ additions in $\mathbb{F}_q$. If one can store $H(z^i)$ for $0 \leq i < m$ then

$$H(u) = \sum_{i=0}^{m-1} u_i H(z^i)$$

can be computed at a cost of $m/2$ additions on average. □

Theorem 6.2. *Let $\mathcal{C}$ and $\mathcal{D}$ be compression and decompression maps, respectively, as described in Theorem 4.4. Then compression via $\mathcal{C}$ requires 1 division in $\mathbb{F}_{q^2}$ and decompression via $\mathcal{D}$ requires $(m-1)/2$ squarings and $(m-1)/2$ additions in $\mathbb{F}_q$, and 1 division in $\mathbb{F}_{q^4}$. If storage for m $\mathbb{F}_q$-elements is available then decompression requires on average $m/2$ additions in $\mathbb{F}_q$ and 1 division in $\mathbb{F}_{q^4}$.*

Proof. The proof follows from Theorem 4.4 and Lemma 6.1. □

6.2. **Exponentiation algorithms.** Recall that in our compression method, given $g = g_0 + g_1\sigma \in G_\ell \setminus \{1\}$, we first compress g to $\alpha = (g_0+1)/g_1 = a + bw$, and then compress α to (i,b) where $i \in \{0,1\}$. By Theorem 6.2, compressing g to (i,b) and decompressing (i,b) to α (and to g) can be achieved at a negligible cost. In this context, we call α a *half-compressed* element.

We present two exponentiation algorithms to compute g^e given $\mathcal{C}(g) = (i,b)$ and $e \in \mathbb{Z}$. The first exponentiation algorithm, which we call *HCTBE* (Half-Compressed Torus-Based Exponentiation), partially decompresses (i,b) to α and then uses a multiplication formula for half-compressed elements. The output is then compressed to obtain $\mathcal{C}(g^e)$. The second algorithm, which we call *FDDE* (Fully-Decompressed Direct Exponentiation Algorithm), fully decompresses (i,b) to g and uses a conventional square-and-multiply exponentiation algorithm in $\mathbb{F}_{q^4}$.

6.2.1. *The HCTBE algorithm.* The algorithm makes use of the multiplication formula (2.3) to compute $\mathcal{C}(g^e)$. The formula requires an inversion in $\mathbb{F}_{q^2}$ that makes the exponentiation algorithm quite costly if one tries to use (2.3) directly. However, the problem can be overcome as follows. If $g = g_0 + g_1\sigma$, $h = h_0 + h_1\sigma \in G_\ell \setminus \{1\}$ are represented by $\alpha = (g_0+1)/g_1$, $\beta = (h_0+1)/h_1 \in \mathbb{F}_{q^2}$, respectively, then we have

$$\begin{aligned} g \cdot h &= \left(\frac{\alpha+\sigma}{\alpha+1+\sigma}\right)\left(\frac{\beta+\sigma}{\beta+1+\sigma}\right) \\ &= \frac{\alpha\beta + c_1 + (\alpha+\beta+1)\sigma}{\alpha\beta + c_1 + \alpha + \beta + 1 + (\alpha+\beta+1)\sigma}. \end{aligned}$$

In other words, if the product of any two elements in G_ℓ is computed by this formula then the result will be of the form

$$\frac{x+y\sigma}{x+y+y\sigma}, \text{ for some } x, y \in \mathbb{F}_{q^2}. \tag{6.1}$$

In particular, given $\mathcal{C}(g) = (i,b)$ and $e \in \mathbb{Z}$, one can first decompress (i,b) to α, and then perform an exponentiation to compute $\mathcal{C}(g^e)$ by using the formulas

$$\begin{aligned} \left(\frac{x+y\sigma}{x+y+y\sigma}\right)^2 &= \frac{x^2 + y^2c_1 + y^2\sigma}{x^2 + y^2c_1 + y^2 + y^2\sigma}, \\ \left(\frac{\alpha+\sigma}{\alpha+1+\sigma}\right)\left(\frac{x+y\sigma}{x+y+y\sigma}\right) &= \frac{\alpha x + yc_1 + (\alpha y + x + y)\sigma}{\alpha x + yc_1 + \alpha y + x + y + (\alpha y + x + y)\sigma}, \end{aligned}$$

in the *square* and *multiply* steps of the exponentiation algorithm. Note that by (6.1) it suffices to only keep track of the numerator during the computations, and to do a single division in $\mathbb{F}_{q^4}$ to obtain g^e and finally its compressed value $\mathcal{C}(g^e)$. Our discussion yields Algorithm 1.

Assuming that $c_1 \in \mathbb{F}_{q^2}$ is chosen so that the cost of multiplying an element by c_1 is negligible, the cost of the squaring step (step 5), and the cost of the multiplication step (step 7) in Algorithm 1 is approximately 2 squarings in $\mathbb{F}_{q^2}$ and 2 multiplications in $\mathbb{F}_{q^2}$, respectively.

6.2.2. *The FDDE algorithm.* After decompressing $\mathcal{C}(g) = (i,b)$ to $g = g_0 + g_1\sigma$, we use a conventional square-and-multiply exponentiation algorithm as described in Algorithm 2. Since

$$\begin{aligned} (x+y\sigma)^2 &= x^2 + y^2c + y^2\sigma, \\ (g_0+g_1\sigma)(x+y\sigma) &= g_0x + g_1yc + (g_0y + g_1x + g_1yc)\sigma, \end{aligned}$$

each squaring step (step 5) in Algorithm 2 requires 2 squarings in $\mathbb{F}_{q^2}$. Using Karatsuba's technique, each multiplication step (steps 7-8) requires 3 multiplications in $\mathbb{F}_{q^2}$. We assume that $c_1 \in \mathbb{F}_{q^2}$ is chosen appropriately so that the cost of multiplying an element by c_1 is negligible.

Algorithm 1 The HCTBE exponentiation algorithm
Input: $\mathcal{C}(g)$ and e
Output: $\mathcal{C}(g^e)$

Write $e = \sum_{i=0}^{s-1} b_i 2^i$ where $b_i \in \{0,1\}$ and $b_{s-1} = 1$
Decompress $\mathcal{C}(g)$ to α by using Theorem 4.4
$x \leftarrow \alpha$, $y \leftarrow 1$
for i from $s-2$ down to 0 **do**
  $y' \leftarrow y^2$, $x' \leftarrow x^2 + y'c_1$
  if $b_i = 1$ **then**
    $x' \leftarrow \alpha x + yc_1$, $y' \leftarrow \alpha y + x + y$
  end if
  $x \leftarrow x'$, $y \leftarrow y'$
end for
$g' \leftarrow (x + y\sigma)/(x + y + y\sigma)$
Compress (g') to $\mathcal{C}(g') = (i', b')$, by using Theorem 4.4
Output (i', b')

Algorithm 2 The FDDE exponentiation algorithm
Input: $\mathcal{C}(g)$ and e
Output: $\mathcal{C}(g^e)$

Write $e = \sum_{i=0}^{s-1} b_i 2^i$ where $b_i \in \{0,1\}$ and $b_{s-1} = 1$
Decompress $\mathcal{C}(g)$ to $g = g_0 + g_1\sigma$ by using Theorem 4.4
$x \leftarrow g_0$, $y \leftarrow g_1$
for i from $s-2$ down to 0 **do**
  $y' \leftarrow y^2$, $x' \leftarrow x^2 + y'c_1$
  if $b_i = 1$ **then**
    $u_0 \leftarrow (g_0 + g_1)(x' + y')$, $u_1 \leftarrow g_0 x'$, $u_2 \leftarrow g_1 y'$, $u_3 \leftarrow u_2 c_1$
    $x' \leftarrow u_1 + u_3$, $y' \leftarrow x' + u_0 + u_2$
  end if
  $x \leftarrow x'$, $y \leftarrow y'$
end for
$g' \leftarrow (x + y\sigma)$
Compress (g') to $\mathcal{C}(g') = (i', b')$, by using Theorem 4.4
Output (i', b')

6.2.3. *A comparison with trace-based exponentiation.* In [8], it was shown that it is possible to compress elements of G_ℓ by a factor 4 by identifying an element $g \in G_\ell$ with its trace $\mathrm{Tr}_{q^4,q}(g)$. Given $\mathrm{Tr}_{q^4,q}(g)$ and an integer e, five exponentiation algorithms were proposed and analyzed in [8] to compute $\mathrm{Tr}_{q^4,q}(g^e)$. The algorithms are based on the following ideas:

(1) Use $\mathrm{Tr}_{q^4,q}(g)$ directly and perform computations in $\mathbb{F}_q$ (Algorithm 1 in [8]).
(2) First decompress $\mathrm{Tr}_{q^4,q}(g)$ to $\mathrm{Tr}_{q^4,q^2}(g)$. Then use $\mathrm{Tr}_{q^4,q^2}(g)$ directly and perform computations in $\mathbb{F}_{q^2}$ (Algorithm 2 in [8]).
(3) First decompress $\mathrm{Tr}_{q^4,q}(g)$ to g and perform computations in $\mathbb{F}_{q^4}$ (Algorithm DDE in [8]).
(4) First decompress $\mathrm{Tr}_{q^4,q}(g)$ to $\mathrm{Tr}_{q^4,q^2}(g)$. Then use $\mathrm{Tr}_{q^4,q^2}(g)$ to construct a copy of $\mathbb{F}_{q^4}$ based on the minimal polynomial of g over $\mathbb{F}_{q^2}$, and perform computations in $\mathbb{F}_{q^4}$ (Algorithm BPV-I in [8]).
(5) Use $\mathrm{Tr}_{q^4,q}(g)$ to construct a copy of $\mathbb{F}_{q^4}$ based on the minimal polynomial of g over $\mathbb{F}_q$, and perform computations in $\mathbb{F}_{q^4}$ (Algorithm BPV-II in [8]).

If a decompression is performed then it is the most expensive step in these algorithms. Therefore, the algorithms based on (1) and (5) are overall faster than the algorithms based on (2), (3) and (4). In particular, Algorithm 1 in [8] was reported to be the fastest exponentiation algorithm in the case of using a general base $\mathrm{Tr}_{q^4,q}(g)$, and its performance was further improved in [7] (see Algorithm 3 in [7]). However, once decompression can be performed in advance, such as in the case of using a fixed base $\mathrm{Tr}_{q^4,q}(g)$, then the algorithm based on (3) is the fastest.

Note that by Theorem 6.2, given $\mathcal{C}(g)$ for some $g \in G_\ell \setminus \{1\}$, one can recover g (and also $\mathrm{Tr}_{q^4,q}(g)$ and $\mathrm{Tr}_{q^4,q^2}(g)$) at a negligible cost. Hence, it is more advantageous to use $\mathcal{C}(g)$ instead of $\mathrm{Tr}_{q^4,q}(g)$. For example, using $\mathcal{C}(g)$, we can obtain faster exponentiation algorithms than the trace-based exponentiation algorithms in the case of a general base by simply computing $\mathrm{Tr}_{q^4,q}(g)$ from $\mathcal{C}(g)$ and adapting an algorithm based on (3).

7. Factor-6 compression and exponentiation algorithms

This section is analogous to Section 6. We analyze the efficiency of the compression and decompression methods proposed in Section 5. We first show that compression and decompression can be achieved at a negligible cost, and then describe two exponentiation algorithms and provide a performance comparison.

7.1. **Compression/decompression costs.** Let $q = 3^m$, m odd, $t = \sqrt{3q}$ and $\ell = q+1-t$. Let $G_\ell \subset \mathbb{F}_{q^6}^*$ be the subgroup with $|G_\ell| = \ell$. Let $\mathbb{F}_{q^3} = \mathbb{F}_q[w]/(w^3-w-1)$ and $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2-c_0)$ where c_0 is a quadratic non-residue in $\mathbb{F}_{q^3}$. We further assume that $\mathbb{F}_q$ is represented as an m-dimensional vector space over $\mathbb{F}_3$ via a polynomial basis $\{1, z, \ldots, z^{m-1}\}$.

Lemma 7.1. *Let $P_6(x) = c^3x^3 + 2c^{2t+3}x + 2(c^{3(t+1)} + c^{2t} + 1) \in \mathbb{F}_q[x]$ for some $c \in \mathbb{F}_q$. If $P_6(x) = 0$ has a solution in $\mathbb{F}_q$ and storage of m $\mathbb{F}_q$-elements is available, then finding the $\mathbb{F}_q$-solutions requires on average $2m/3$ additions, 2 multiplications, 1 squaring and 1 division in $\mathbb{F}_q$.*

Proof. First observe that if $B \in \mathbb{F}_q$ is a quadratic non-residue and $x^3 + Bx + C = 0$ has a solution in $\mathbb{F}_q$ then all solutions are given by

$$\{r_1, r_2, r_3\} = \{(-B)^{1/2}R(D), (-B)^{1/2}(R(D)+1), (-B)^{1/2}(R(D)+2)\},$$

where $D = C/(-B)^{3/2}$ and $R(D)$ is a root of

$$x^3 - x + D.$$

Clearly, if $x^3 - x + D = 0$ has a solution $R(D) \in \mathbb{F}_q$ then it can be found trit-wise when a normal basis $\{\theta, \theta^3, \ldots, \theta^{3^{m-1}}\}$ is used to represent $\mathbb{F}_q$ as an m-dimensional vector space over $\mathbb{F}_3$. Let us suppose that $R(D) = \sum_{i=0}^{m-1} R_i\theta^{3^i}$ and the m $\mathbb{F}_q$-elements

$$\theta^{3^i} = \sum_{j=0}^{m-1} \theta_{ij} z^j,\ 0 \le i < m$$

are precomputed and stored. Then a solution

$$R(D) = \sum_{i=0}^{m-1} R_i \sum_{j=0}^{m-1} \theta_{ij} z^j$$

to $x^3 - x + D = 0$ is obtained in $\mathbb{F}_q$, at an average cost of $2m/3$ additions in $\mathbb{F}_q$.

Now, in order to find a solution of $P_6(x) = c^3x^3 + 2c^{2t+3}x + 2(c^{3(t+1)} + c^{2t} + 1) = 0$ in $\mathbb{F}_q$, we first compute $B = 2c^{2t+3}/c^3 = 2c^{2t}$ and $C = 2(c^{3(t+1)} + c^{2t} + 1)/c^3$. Then

$$D = C/(-B)^{3/2} = (2(c^{3(t+1)} + c^{2t} + 1)/c^{3t+3})$$

can be computed at a cost of 1 multiplication, 1 squaring and 1 division in $\mathbb{F}_q$ (we ignore the cost of addition in $\mathbb{F}_q$ and Frobenius operations). From our argument above we can find a solution of $x^3 - x + D = 0$ in $\mathbb{F}_q$ at an average cost of $2m/3$ additions in $\mathbb{F}_q$. Hence, the solutions of $P_6(x) = 0$ are given by

$$\{r_1, r_2, r_3\} = \{c^t R(D), c^t(R(D)+1), c^t(R(D)+2)\},$$

and can be obtained at an average cost of $2m/3$ additions, 2 multiplications, 1 squaring and 1 division in $\mathbb{F}_q$. □

From now on, we shall assume that $m \equiv 5 \pmod{12}$ and $c_0 = -1$.

Theorem 7.2. *Let $\mathcal{C}$ and $\mathcal{D}$ be compression and decompression maps, respectively, as described in Theorem 5.4. Then compression via $\mathcal{C}$ requires 1 division in $\mathbb{F}_{q^3}$ and decompression via $\mathcal{D}$ requires on average $2m/3$ additions, 2 multiplications, 1 squaring, 1 division in $\mathbb{F}_q$, and 1 division in $\mathbb{F}_{q^6}$, with a storage of m $\mathbb{F}_q$-elements.*

Proof. The proof follows from Theorem 5.4 and Lemma 7.1. □

7.2. **Exponentiation algorithms.** Recall that in our compression method, given $g = g_0 + g_1\sigma \in G_\ell \setminus \{\pm 1\}$, we first compress g to $\alpha = (g_0 + 1/g_1) = a + bw + cw^2$, by a factor 2, and then compress α to (i, c), $i \in \{0, 1, 2\}$ by a factor of 3. By Theorem 7.2, compressing g to (i, c), and decompressing (i, c) to α (and to g) can be achieved at a negligible cost. In this context, we call α a *half-compressed* element.

We present two exponentiation algorithms to compute g^e given $\mathcal{C}(g) = (i, c)$ and $e \in \mathbb{Z}$. The first exponentiation algorithm, which we call *HCTBE* (Half-Compressed Torus-Based Exponentiation), partially decompresses (i, c) to α and uses a multiplication formula for half-compressed elements. The output is then compressed to obtain $\mathcal{C}(g^e)$. The second algorithm, which we call *FDDE* (Fully-Decompressed Direct Exponentiation Algorithm), fully decompresses (i, c) to g and uses a conventional cube-and-multiply exponentiation algorithm in $\mathbb{F}_{q^6}$.

7.2.1. *The HCTBE algorithm.* The algorithm makes use of the multiplication formula (2.4) to compute $\mathcal{C}(g^e)$. If $g = g_0 + g_1\sigma$, $h = h_0 + h_1\sigma \in G_\ell \setminus \{\pm 1\}$ are represented by $\alpha = (g_0 + 1)/g_1$, $\beta = (h_0 + 1)/h_1 \in \mathbb{F}_{q^3}$, respectively, then we have

$$\begin{aligned} g \cdot h &= \left(\frac{\alpha + \sigma}{\alpha - \sigma}\right)\left(\frac{\beta + \sigma}{\beta - \sigma}\right) \\ &= \frac{\alpha\beta + c_0 + (\alpha + \beta)\sigma}{\alpha\beta + c_0 - (\alpha + \beta)\sigma}. \end{aligned}$$

In other words, if the product of any two elements in G_ℓ is computed by this formula then the result will be of the form

$$\frac{x + y\sigma}{x - y\sigma}, \text{ for some } x, y \in \mathbb{F}_{q^3}. \tag{7.1}$$

In particular, given $\mathcal{C}(g) = (i, c)$ and $e \in \mathbb{Z}$, one can first decompress (i, c) to α, and then perform an exponentiation to compute $\mathcal{C}(g^e)$ by using the formulas

$$\begin{aligned} \left(\frac{x + y\sigma}{x - y\sigma}\right)^3 &= \frac{x^3 + y^3 c_0 \sigma}{x^3 - y^3 c_0 \sigma}, \\ \left(\frac{\alpha + \sigma}{\alpha - \sigma}\right)\left(\frac{x + y\sigma}{x - y\sigma}\right) &= \frac{\alpha x + y c_0 + (\alpha y + x)\sigma}{\alpha x + y c_0 - (\alpha y + x)\sigma}, \\ \left(\frac{\alpha - \sigma}{\alpha + \sigma}\right)\left(\frac{x + y\sigma}{x - y\sigma}\right) &= \frac{\alpha x - y c_0 + (\alpha y - x)\sigma}{\alpha x - y c_0 - (\alpha y - x)\sigma} \end{aligned}$$

in the *cube* and *multiply* steps of the exponentiation algorithm. Note that by (7.1) it suffices to only keep track of the numerator during the computations, and to do a single division in $\mathbb{F}_{q^6}$ to obtain g^e and finally its compressed value $\mathcal{C}(g^e)$. Our discussion yields Algorithm 3.

Algorithm 3 The HCTBE exponentiation algorithm
Input: $\mathcal{C}(g)$ and e
Output: $\mathcal{C}(g^e)$

1: Write $e = \sum_{i=0}^{s-1} b_i 3^i$ where $b_i \in \{-1, 0, 1\}$ and $b_{s-1} = 1$
2: Decompress $\mathcal{C}(g)$ to α by using Theorem 5.4
3: $x \leftarrow \alpha$, $y \leftarrow 1$
4: **for** i from $s-2$ down to 0 **do**
5: $\quad x' \leftarrow x^3$, $y' \leftarrow y^3 c_0$
6: $\quad$ **if** $b_i = 1$ **then**
7: $\quad\quad x' \leftarrow \alpha x + y c_0$, $y' \leftarrow (\alpha y + x)$
8: $\quad$ **else if** $b_i = -1$ **then**
9: $\quad\quad x' \leftarrow \alpha x - y c_0$, $y' \leftarrow (\alpha y - x)$
10: $\quad$ **end if**
11: $\quad x \leftarrow x'$, $y \leftarrow y'$
12: **end for**
13: $g' \leftarrow (x + y\sigma)/(x - y\sigma)$
14: Compress (g') to $\mathcal{C}(g') = (i', c')$, by using Theorem 5.4
15: Output (i', c')

Since $c_0 = -1$, the cost of the cubing step (step 5) and the cost of the multiplication step (step 7 or step 9) in Algorithm 3 is approximately 2 cubings in $\mathbb{F}_{q^3}$ and 2 multiplications in $\mathbb{F}_{q^3}$, respectively.

Remark 7.3. Granger, Page and Stam [5, Section 3.2] proposed an exponentiation algorithm that works in the quotient group $\mathbb{F}_{q^6}^*/\mathbb{F}_{q^3}^*$ where $q = 3^m$, m is odd, and mimics the mixed addition method for point multiplication on elliptic curves. Algorithm 1 can be seen as analogous to their algorithm. The main difference is that they identify $g = g_0 + g_1\sigma$ with $\alpha = g_0/g_1$ instead of $\alpha = (g_0 + 1)/g_1$ and therefore their method cannot be directly adapted to obtain a fast exponentiation algorithm in $G_\ell \subset \mathbb{F}_{q^6}^*$. In particular, it was reported in [5, Table 3] that exponentiation in $\mathbb{F}_{q^6}^*/\mathbb{F}_{q^3}^*$ is more efficient than exponentiation in G_ℓ. The HCTBE algorithm equalizes the efficiency of exponentiation algorithms in G_ℓ and $\mathbb{F}_{q^6}^*/\mathbb{F}_{q^3}^*$.

7.2.2. *The FDDE algorithm.* After decompressing $\mathcal{C}(g) = (i, b)$ to $g = g_0 + g_1\sigma$, we use a conventional cube-and-multiply exponentiation algorithm as described in Algorithm 4. Since

$$\begin{aligned}(x + y\sigma)^3 &= x^3 + y^3 c_0 \sigma,\\ (g_0 + g_1\sigma)(x + y\sigma) &= g_0 x + g_1 y c_0 + (g_0 y + g_1 x)\sigma,\\ (g_0 - g_1\sigma)(x + y\sigma) &= g_0 x - g_1 y c_0 + (g_0 y - g_1 x)\sigma,\end{aligned}$$

each cubing step (step 5) in Algorithm 4 requires 2 cubings in $\mathbb{F}_{q^3}$. Using Karatsuba's technique, each multiplication step (steps 7-8 or steps 10-11) requires 3 multiplications in $\mathbb{F}_{q^3}$ (note that $c_0 = -1$).

7.2.3. *A comparison with trace-based exponentiation.* In [15], it was shown that it is possible to compress elements of G_ℓ by a factor 6 by identifying an element $g \in G_\ell$ with its trace $\mathrm{Tr}_{q^6,q}(g)$. Given $\mathrm{Tr}_{q^6,q}(g)$ and an integer e, six exponentiation algorithms were proposed and analyzed in [8] to compute $\mathrm{Tr}_{q^6,q}(g^e)$. The performance of these six algorithms were also compared with a previously-known exponentiation algorithm XTR_3 in [15]. The algorithms are based on the following ideas:

Algorithm 4 The FDDE exponentiation algorithm
Input: $\mathcal{C}(g)$ and e
Output: $\mathcal{C}(g^e)$

Write $e = \sum_{i=0}^{s-1} b_i 3^i$ where $b_i \in \{-1, 0, 1\}$ and $b_{s-1} = 1$
Decompress $\mathcal{C}(g)$ to $g = g_0 + g_1\sigma$ by using Theorem 5.4
$x \leftarrow g_0$, $y \leftarrow g_1$
for i from $s-2$ down to 0 **do**
  $x' \leftarrow x^3$, $y' \leftarrow y^3 c_0$
  if $b_i = 1$ **then**
    $u_0 \leftarrow (g_0 + g_1)(x' + y')$, $u_1 \leftarrow g_0 x'$, $u_2 \leftarrow g_1 y'$, $u_3 \leftarrow u_2 c_0$
    $x' \leftarrow u_1 + u_3$, $y' \leftarrow u_0 - (u_1 + u_2)$
  else if $b_i = -1$ **then**
    $u_0 \leftarrow (g_0 - g_1)(x' + y')$, $u_1 \leftarrow g_0 x'$, $u_2 \leftarrow -g_1 y'$, $u_3 \leftarrow u_2 c_0$
    $x' \leftarrow u_1 + u_3$, $y' \leftarrow u_0 - (u_1 + u_2)$
  end if
  $x \leftarrow x'$, $y \leftarrow y'$
end for
$g' \leftarrow (x + y\sigma)$
Compress (g') to $\mathcal{C}(g') = (i', c')$, by using Theorem 5.4
Output (i', c')

(1) Use $\mathrm{Tr}_{q^6,q}(g)$ directly and perform computations in $\mathbb{F}_q$ (Algorithm 3 in [8]).
(2) First decompress $\mathrm{Tr}_{q^6,q}(g)$ to $\mathrm{Tr}_{q^6,q^3}(g)$. Then use $\mathrm{Tr}_{q^6,q^3}(g)$ directly and perform computations in $\mathbb{F}_{q^3}$ (Algorithm 4 in [8]).
(3) First decompress $\mathrm{Tr}_{q^6,q}(g)$ to g and perform computations in $\mathbb{F}_{q^6}$ (Algorithm DDE in [8]).
(4) First decompress $\mathrm{Tr}_{q^6,q}(g)$ to $\mathrm{Tr}_{q^6,q^2}(g)$. Then use $\mathrm{Tr}_{q^6,q^2}(g)$ to construct a copy of $\mathbb{F}_{q^6}$ based on the minimal polynomial of g over $\mathbb{F}_{q^2}$, and perform computations in $\mathbb{F}_{q^6}$ (Algorithm BPV-I in [8]).
(5) First decompress $\mathrm{Tr}_{q^6,q}(g)$ to $\mathrm{Tr}_{q^6,q^3}(g)$. Then use $\mathrm{Tr}_{q^6,q^3}(g)$ to construct a copy of $\mathbb{F}_{q^6}$ based on the minimal polynomial of g over $\mathbb{F}_{q^3}$, and perform computations in $\mathbb{F}_{q^6}$ (Algorithm BPV-II in [8]).
(6) Use $\mathrm{Tr}_{q^6,q}(g)$ to construct a copy of $\mathbb{F}_{q^6}$ based on the minimal polynomial of g over $\mathbb{F}_q$, and perform computations in $\mathbb{F}_{q^6}$ (Algorithm BPV-III in [8]).
(7) First decompress $\mathrm{Tr}_{q^6,q}(g)$ to $\mathrm{Tr}_{q^6,q^2}(g)$. Then use $\mathrm{Tr}_{q^6,q^2}(g)$ directly and perform computations in $\mathbb{F}_{q^2}$ (Algorithm XTR_3 in [15]).

The algorithms based on (1), (4), (6) and (7) are overall faster than the algorithms based on (2), (3) and (5) because of the expensive decompression operations required in the latter algorithms. In particular, it was reported in [8] that XTR_3 in [15] can be further sped up and it is the fastest exponentiation algorithm for general bases. However, if decompression can be precomputed, for example when the base is fixed, then the algorithm based on (3) is the fastest.

Note that by Theorem 7.2, given $\mathcal{C}(g)$ for some $g \in G_\ell \setminus \{\pm 1\}$, one can recover g (and also $\mathrm{Tr}_{q^6,q}(g)$, $\mathrm{Tr}_{q^6,q^2}(g)$ and $\mathrm{Tr}_{q^6,q^3}(g)$) at a negligible cost. Hence, it is more advantageous to use $\mathcal{C}(g)$ instead of $\mathrm{Tr}_{q^6,q}(g)$. For example, using $\mathcal{C}(g)$, we can obtain faster exponentiation algorithms than the trace-based exponentiation algorithms in the case of a general base $\mathrm{Tr}_{q^6,q}(g)$, by simply computing $\mathrm{Tr}_{q^6,q}(g)$ from $\mathcal{C}(g)$ and adapting an algorithm based on (3).

8. A comparison of exponentiation algorithms

In this section, we estimate the running times of the exponentiation algorithms discussed in Sections 6.2 and 7.2, and compare them with the fastest previously-known exponentiation algorithms. We consider the case of a general base, $\mathcal{C}(g)$ or $\mathrm{Tr}_{q^k,q}(g)$, which is the most interesting case because when the base is fixed we may ignore the cost of obtaining one of $\mathcal{C}(g)$ and $\mathrm{Tr}_{q^k,q}(g)$ from the other, and hence obtain an equivalent performance in torus-based and trace-based exponentiation algorithms.

We denote by C_i, M_i, and S_i the operations of cubing, multiplication, and squaring in $\mathbb{F}_{q^i}$ for $i = 1, 2, 3$. We assume that $S_2 = 2S_1$ for characteristic two, $C_3 = 3C_1$ for characteristic three, and also assume, using Karatsuba's technique, that $M_2 = 3M_1$ and $M_3 = 6M_1$.

Note that the HCTBE and FDDE algorithms can easily be modified to work with window NAF techniques. In particular, we assume that the width-w radix-2 and radix-3 NAF representation of the exponent e are used in for the characteristic-two and the characteristic-three cases, respectively. Note that width-w radix-2 and radix-3 NAF representations of e contain on average $\log_2 e/(w+1)$ and $2\log_3 e/(2w+1)$ nonzero digits, respectively; see for example [18].

The estimated costs of the exponentiation algorithms are presented in Table 1. In our analysis, we ignore the compression/decompression costs and also the precomputation costs required for window NAF methods as they are negligible comparing to the overall cost of algorithms.

Table 1. Comparison of exponentiation algorithms for factor-4 and factor-6 compression in the case of a general base. The exponent is e.

Algorithms	Main Loop
Characteristic-two fields	
Algorithm 3 in [7]	$(3.19M_1)\log_2 e$
FDDE	$(4S_1 + \frac{9}{(w+1)}M_1)\log_2 e$
HCTBE	$(4S_1 + \frac{6}{(w+1)}M_1)\log_2 e$
Characteristic-three fields	
XTR_3 in [15]	$(3M_1)\log_2 e$
FDDE	$(6C_1 + \frac{36}{(2w+1)}M_1)\log_3 e$
HCTBE	$(6C_1 + \frac{24}{(2w+1)}M_1)\log_3 e$

Assuming that S_1 and C_1 are essentially free in characteristic-two and characteristic-three fields, respectively, and setting $w = 3$, we can estimate the cost of FDDE as $(2.25M_1)\log_2 e$, and the cost of HCTBE as $(1.5M_1)\log_2 e$ in characteristic-two fields. Similarly, the cost of FDDE and HCTBE in characteristic-three fields can be approximated as $(3.24M_1)\log_2 e$ and $(2.16M_1)\log_2 e$, respectively.

Therefore, if we require that the input to an exponentiation algorithm and the output of the algorithm are the compressed representation of g and g^e, it seems best to compress g to $\mathcal{C}(g)$ by a factor of 4 or 6, and to use the HCTBE algorithms to compute the factor-4 or factor-6 compressed representation $\mathcal{C}(g^e)$ of g^e. It also seems that the HCTBE algorithms outperform the fastest previously-known exponentiation algorithms in G_ℓ. The reason is that compression/decompression costs in the HCTBE algorithms are negligible and that each multiplication step in the HCTBE algorithm in characteristic-two requires $6M_1$ whereas it would require $9M_1$ in a conventional exponentiation algorithm adapting Karatsuba's method. Similarly, each multiplication step in the HCTBE algorithm in characteristic-three requires $12M_1$ whereas it would require $18M_1$ in a conventional exponentiation algorithm adapting Karatsuba's method (see also Remark 7.3).

To be more concrete, we list the expected running times of the six exponentiation algorithms in a particular setting in Table 2 based on the estimates given in Table 1. For the 128-bit security level, in the characteristic-two case we let $q = 2^{1223}$ and $t = 2^{612}$. Then $q + 1 + t = 5\ell$ where ℓ is a 1221-bit prime. We will ignore the cost S_1. In the characteristic-three case, we let $q = 3^{509}$ and $t = 3^{255}$. Then $q + 1 - t = 7\ell$ where ℓ is an 804-bit prime. We will ignore the cost C_1. In both cases, we choose $w = 3$.

TABLE 2. Comparison of exponentiation algorithms for factor-4 and factor-6 compression in the case of a general base at the 128-bit security level. The exponent is an 1221-bit integer in the characteristic-two case, and an 804-bit integer in the characteristic-three case.

Algorithms	Main Loop
A characteristic-two field	
Algorithm 3 in [7]	$3895M_1$
FDDE	$2747M_1$
HCTBE	$1831M_1$
A characteristic-three field	
XTR_3 in [15]	$2412M_1$
FDDE	$2609M_1$
HCTBE	$1739M_1$

9. Concluding remarks

We showed that by building on torus-based compression techniques, it is possible to compress elements in G_ℓ by a factor of 4 when $|G_\ell| = \ell = q + 1 \pm t$, $q = 2^m$ and $t = \sqrt{2q}$; and by a factor of 6 when $|G_\ell| = \ell = q + 1 - t$, $q = 3^m$ and $t = \sqrt{3q}$. Our methods achieve the best possible compression ratio in G_ℓ, and moreover have the feature that the compression and decompression maps are computable at a negligible cost. We discussed several exponentiation algorithms and, in particular, showed that HCTBE outperforms the fastest exponentiation algorithms in both the characteristic-two and the characteristic-three cases.

We note that the pairing values of bilinear pairings derived from supersingular elliptic curves of embedding degrees 4 and 6 over finite fields of characteristic two and three, and derived from supersingular hyperelliptic curves of embedding degrees 12 over finite fields of characteristic two, lie in G_ℓ for a suitable choice of parameters. Therefore, our techniques can be easily incorporated into pairing-based protocols that require exponentiations or products of pairings; examples of such protocols include Scott's identity-based key agreement protocol [14] and Waters signature scheme [21].

Our compression method compresses $g \in G_\ell$ to an element $\mathcal{C}(g)$ in $\mathbb{F}_q$. However, given $\mathcal{C}(g)$ and $e \in \mathbb{Z}$, all the exponentiation algorithms to compute $\mathcal{C}(g^e)$ first decompresses $\mathcal{C}(g)$ (at least partially), and then exponentiate. It is natural to ask if one can devise a multiplication formula for $g, h \in G_\ell$ which computes $\mathcal{C}(g) * \mathcal{C}(h) = \mathcal{C}(gh)$ directly in $\mathbb{F}_q$.

Acknowledgment

The author would like to thank Alfred Menezes for his careful reading of the earlier drafts of this paper and for his corrections and suggestions.

References

[1] É. Brier and M. Joye. Weirstraß elliptic curves and side channel attacks. *Public Key Cryptography – PKC 2002, Lecture Notes In Computer Science*, 2274:335–345, 2002.

[2] A. Brouwer, R. Pellikaan, and E. Verheul. Doing more with fewer bits. *Advances in Cryptology – ASIACRYPT '99, Lecture Notes in Computer Science*, 1716:321–332, 1999.

[3] K. Giuliani and G. Gong. Analogues to the Gong-Harn and XTR cryptosystems. *Technical Report CORR 2003-34, University of Waterloo*, 2003. Available at `http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-34.ps`.

[4] G. Gong and L. Harn. Public-key cryptosystems based on cubic finite field extensions. *IEEE Transactions on Information Theory*, 45:2601–2605, 1999.

[5] R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing-based cryptography. *LMS Journal of Computation and Mathematics*, 9:64–85, 2004.

[6] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to elliptic curve cryptography*. Springer-Verlag, New York, USA, 2004.

[7] K. Karabina. Double-exponentiation in factor-4 groups and its applications. *Twelfth IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science*, 5921:336–350, 2009.

[8] K. Karabina. Factor-4 and 6 compression of cyclotomic subgroups of $\mathbb{F}_{2^{4m}}^*$ and $\mathbb{F}_{3^{6m}}^*$. *Journal of Mathematical Cryptology*, 4:1–42, 2010.

[9] A. Lenstra and E. Verheul. The XTR public key system. *Advances in Cryptology – CRYPTO 2000, Lecture Notes in Computer Science*, 1880:1–19, 2000.

[10] J. López and R. Dahab. Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation. *Cryptographic Hardware and Embedded Systems, Lecture Notes In Computer Science*, 1717:316–327, 1999.

[11] P. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243–264, 1987.

[12] K. Rubin and A. Silverberg. Torus-based cryptography. *Advances in Cryptology – CRYPTO 2003, Lecture Notes in Computer Science*, 2729:349–365, 2003.

[13] K. Rubin and A. Silverberg. Compression in finite fields and torus-based cryptography. *SIAM Journal on Computing*, 37:1401–1428, 2008.

[14] M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/2002/164`.

[15] M. Shirase, D. Han, Y. Hibin, H. Kim, and T. Takagi. A more compact representation of XTR cryptosystem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E91-A:2843–2850, 2008.

[16] 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 – ASIACRYPT '94, Lecture Notes In Computer Science*, 917:357–364, 1994.

[17] M. Stam and A. Lenstra. Speeding up XTR. *Advances in Cryptology – ASIACRYPT 2001, Lecture Notes in Computer Science*, 2248:125–143, 2001.

[18] T. Takagi, S. Yen, and B. Wu. Radix-r non-adjacent form. *Information Security – ISC 2004, Lecture Notes In Computer Science*, 3225:99–110, 2004.

[19] M. van Dijk, R. Granger, D. Page, K. Rubin, A. Silverberg, M. Stam, and D. Woodruff. Practical cryptography in high dimensional tori. *Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science*, 3494:234–250, 2005.

[20] M. van Dijk and D. Woodruff. Asymptotically optimal communication for torus-based cryptography. *Advances in Cryptology – CRYPTO 2004, Lecture Notes in Computer Science*, 3152:151–178, 2004.

[21] B. Waters. Efficient identity-based encryption without random oracles. *Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science*, 3494:114–127, 2005.

Dept. of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

E-mail address: `kkarabin@uwaterloo.ca`