FACTOR-4 AND 6 COMPRESSION OF CYCLOTOMIC SUBGROUPS OF $\mathbb{F}^*_{2^{4m}}$ AND $\mathbb{F}^*_{3^{6m}}$

KORAY KARABINA

Abstract. Bilinear pairings derived from supersingular elliptic curves of embedding degrees 4 and 6 over finite fields $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$, respectively, have been used to implement pairing-based cryptographic protocols. The pairing values lie in certain prime-order subgroups of the cyclotomic subgroups of orders $2^{2m}+1$ and $3^{2m}-3^m+1$, respectively, of the multiplicative groups $\mathbb{F}^*_{2^{4m}}$ and $\mathbb{F}^*_{3^{6m}}$. It was previously known how to compress the pairing values over characteristic two fields by a factor of 2, and the pairing values over characteristic three fields by a factor of 6. In this paper, we show how the pairing values over characteristic two fields can be compressed by a factor of 4. Moreover, we present and compare several algorithms for performing exponentiation in the prime-order subgroups using the compressed representations. In particular, in the case where the base is fixed, we expect to gain at least a 54% speed up over the fastest previously known exponentiation algorithm that uses factor-6 compressed representations.

1. Introduction

The Diffie-Hellman key agreement protocol [8] can be used by two parties A and B to establish a shared secret by communicating over an unsecured channel. Let $G = \langle g \rangle$ be a prime-order subgroup of the multiplicative group $\mathbb{F}^*_q$ of a finite field $\mathbb{F}_q$. Party A selects a private key a and sends g^a to B. Similarly, B selects a private key b and sends g^b to A. Both parties can then compute the shared secret g^{ab}. Security of the protocol depends on the intractability of the problem of computing g^{ab} from g^a and g^b; this is called the Diffie-Hellman problem in G. The best method known for solving the Diffie-Hellman problem in G is to solve the discrete logarithm problem in G, that is, computing a from g^a. If q is prime (say $q = p$), then the fastest algorithms known for solving the discrete logarithm problem in G are Pollard's rho method [20] and the number field sieve [12]. To achieve a 128-bit security level against these attacks, one needs to select $\#G \approx 2^{256}$ and $p \approx 2^{3072}$ [9, Section 4.2]. Note that even though the order of G is approximately 2^{256}, the natural representation of elements of G, namely as integers modulo p, are approximately 3072 bits in length. This brings an overhead both to the efficiency of the protocol and to the number of bits that need to be stored or transmitted. In recent years, there have been several proposals for compressing the elements of certain subgroups of certain finite fields.

Date: June 23, 2009; revised on April 27, 2010.

Key words and phrases. Finite field compression, cyclotomic subgroups, pairing-based cryptography.

The first proposal was by Smith and Skinner in 1994 [27] (see also [4]). The main idea is that Lucas functions can be used modulo a prime to perform exponentiation in cryptographic applications. In fact, using this method the elements of the order-$(q+1)$ subgroup G of $\mathbb{F}_{q^2}^*$ can be identified by their traces over $\mathbb{F}_q$. More precisely, the elements of G can be uniquely identified up to conjugation over $\mathbb{F}_q$. This construction yields a compression factor of 2.

Gong and Harn [11] obtained a factor-3/2 compression and efficient exponentiation for the compressed form of the elements in the order-(p^2+p+1) subgroup G of $\mathbb{F}_{p^3}^*$. Elements of G are represented by a pair of elements from $\mathbb{F}_p$. Similarly, Giuliani and Gong [10] obtained a factor-5/2 compression and efficient exponentiation for the compressed form of the elements in the order-$(p^4-p^3+p^2-p+1)$ subgroup G of $\mathbb{F}_{p^{10}}^*$. Brouwer, Pellikaan and Verheul [6] obtained a factor-3 compression by representing elements of the order-(p^2-p+1) subgroup G of $\mathbb{F}_{p^6}^*$ by a pair of elements from $\mathbb{F}_p$. Even though they did not give an algorithm to exponentiate the elements in G in their compressed form, they noted that to exponentiate an element in G it suffices to know its compressed form and the exponent. In 2000, Lenstra and Verheul [17] showed that elements of the order-(p^2-p+1) subgroup G of $\mathbb{F}_{p^6}^*$ can be uniquely represented (up to conjugation over $\mathbb{F}_{p^2}$) by their traces over $\mathbb{F}_{p^2}$. Note that the compression factor is the same as in [6]. An important contribution of Lenstra and Verheul was a very efficient algorithm for exponentiation in G using the trace representation. More recently, Shirase et al. [25] observed that the elements of the order-$(q-\sqrt{3q}+1)$ subgroup G of $\mathbb{F}_{q^6}^*$, where $q=3^m$ for some odd number m, can be uniquely represented (up to conjugation over $\mathbb{F}_q$) by their traces over $\mathbb{F}_q$, thereby achieving a factor-6 compression. They also presented an algorithm for exponentiation in G. If $g \in G$ and c is its factor-6 compressed form in $\mathbb{F}_q$, they first lifted c to the trace of g in $\mathbb{F}_{q^2}$ and thereafter used an analogue of the Lenstra-Verheul algorithm to exponentiate g.

Rubin and Silverberg [21] introduced a compression/decompression method for finite field elements by using a rational parametrization of an algebraic torus. For a positive integer k and a prime power q the algebraic torus $\mathbb{T}_k$ is a $\varphi(k)$-dimensional algebraic variety over $\mathbb{F}_q$ and its group $\mathbb{T}_k(\mathbb{F}_q)$ of $\mathbb{F}_q$-rational points is isomorphic to the order-$\Phi_k(q)$ 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. Rubin and Silverberg noted that one would hope to use only $\varphi(k)$ elements in $\mathbb{F}_q$ in order to (uniquely) represent elements of $\mathbb{T}_k(\mathbb{F}_q)$. For the cases $k=2$ and $k=6$, they presented explicit compression/decompression algorithms for the elements of $\mathbb{T}_k(\mathbb{F}_q)$, and showed that the Smith-Skinner, Gong-Harn and Lenstra-Verheul representations are based on certain quotients of the algebraic tori, $\mathbb{T}_2, \mathbb{T}_3$ and $\mathbb{T}_6$, respectively, thus explaining the compression ratios of $2/\varphi(2)=2$, $3/\varphi(3)=3/2$ and $6/\varphi(6)=3$. Later, van Dijk et al. [30], improving on an earlier work [31], constructed an efficient bijection between $\mathbb{T}_k(\mathbb{F}_q) \times \mathbb{F}_q^m$ and $\mathbb{F}_q^{\varphi(k)+m}$ and obtained, asymptotically, a compression factor of $k/\varphi(k)$. In particular, when representing i elements in $\mathbb{T}_k(\mathbb{F}_q)$ for $k=30$ and $k=210$, they obtained compression factors $30i/(8i+2)$ with $m=2$, and $210i/(48i+24)$ with $m=24$, respectively.

We provide more details about some of the previous work in Appendix A.

Let $q = 2^m$, where m is odd, and note that

$$\begin{aligned} q^4 - 1 &= (q-1)(q+1)\Phi_4(q) \\ &= (q-1)(q+1)(q-\sqrt{2q}+1)(q+\sqrt{2q}+1). \end{aligned}$$

In this paper, we achieve a factor-4 compression for the subgroups G of $\mathbb{T}_4(\mathbb{F}_q)$ of orders $q \pm \sqrt{2q}+1$. We show that the elements of G can be uniquely represented (up to conjugation over $\mathbb{F}_q$) using their traces over $\mathbb{F}_q$, and that exponentiation in G can be efficiently performed using the compressed representations. Our method gives a better compression factor than the Smith-Skinner system. We note that our factor-4 compression does not contradict the Rubin-Silverberg observation about the necessity of using $\varphi(k)$ $\mathbb{F}_q$-elements for representing elements of $\mathbb{T}_k(\mathbb{F}_q)$ since our construction compresses elements of subgroups of relatively small order of $\mathbb{T}_4(\mathbb{F}_q)$ and the resulting set of compressed elements do not preserve the group structure.

Let $q = 3^m$, where m is odd, and note that

$$\begin{aligned} q^6 - 1 &= (q^3-1)(q+1)\Phi_6(q) \\ &= (q^3-1)(q+1)(q-\sqrt{3q}+1)(q+\sqrt{3q}+1). \end{aligned}$$

As mentioned earlier, Shirase et al. showed that by using traces over $\mathbb{F}_q$ one can achieve a factor-6 compression (up to conjugation over $\mathbb{F}_q$) for the elements of the order-$(q-\sqrt{3q}+1)$ subgroup G of $\mathbb{T}_6(\mathbb{F}_q)$. Moreover, exponentiation in G can be efficiently performed when elements are represented by their traces over $\mathbb{F}_{q^2}$. We observe that a similar compression technique and efficient exponentiation also applies to the order-$(q+\sqrt{3q}+1)$ subgroup G of $\mathbb{T}_6(\mathbb{F}_q)$. Suppose that $g \in G$ and $c \in \mathbb{F}_q$ is its factor-6 compressed representation. We present six exponentiation algorithms. The first works directly with the compressed element c. The second algorithm first lifts c to the trace of g over $\mathbb{F}_{q^3}$, and then employs an exponentiation algorithm of Scott and Barreto [24]. The third algorithm first lifts c to g, and then uses a conventional exponentiation method. In the fourth algorithm, we first determine $f_2(x)$, the minimal polynomial of g over $\mathbb{F}_{q^2}$, by partially decompressing c to an element in $\mathbb{F}_{q^2}$. Then we construct $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[x]/(f_2(x))$, and use a conventional exponentiation method. The idea of the fifth and sixth algorithms is similar to the fourth algorithm except that we use the minimal polynomials of g over $\mathbb{F}_{q^3}$ and $\mathbb{F}_q$, respectively. In the case where the base is fixed, the first algorithm is expected to be at least 54% faster than the XTR$_3$ algorithm presented in [25].

Besides reducing transmission costs in the Diffie-Hellman and related protocols, we observe that compression techniques have applications in pairing-based cryptography where bilinear pairings derived from supersingular elliptic curves of embedding degree 4 and 6 over finite fields $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$ are employed. The pairing values lie in prime-order subgroups of orders dividing $q \pm \sqrt{2q}+1$ and $q \pm \sqrt{3q}+1$ (where $q = 2^m$ or $q = 3^m$) of $\mathbb{F}^*_{2^{4m}}$ and $\mathbb{F}^*_{3^{6m}}$, respectively, and thus it can be beneficial to compress these pairing values.

The remainder of the paper is organized as follows. We begin in Section 2 by describing some cryptographic applications of our compression methods. Section 3 introduces some terminology and sets the notation that we will use throughout the paper. Sections 4 and 5 describe our compression and exponentiation techniques for the characteristic two and three fields. The exponentiation methods are compared in Section 6. We make some concluding remarks in Section 7.

2. Cryptographic applications

In this section we give some examples of cryptographic protocols where our compression techniques can be beneficial.

As described in Section 1, compression is useful in Diffie-Hellman and related key agreement protocols where the underlying group is a prime-order subgroup of the multiplicative group of a finite field. Indeed, Koblitz [15] studied the efficiency of discrete logarithm protocols when the underlying group G is a subgroup of $\mathbb{T}_6(\mathbb{F}_q)$ for $q = 3^m$, and where $\#G$ divides $q \pm \sqrt{3q} + 1$.

Beginning with the seminal work of Joux [14], Sakai-Ohgishi-Kasahara [22] and Boneh-Franklin [5], bilinear pairings have been widely used to design protocols for various cryptographic tasks. These protocols can be described using symmetric bilinear pairings $e : G_1 \times G_1 \to G$ where G_1 and G are groups of prime order n. A necessary condition for the security of these protocols is that the discrete logarithm problems in G_1 and G should be intractable. Such pairings can be realized by selecting G_1 to be a group of points in $E_1(\mathbb{F}_q)$, where $q = 2^m$ and $E_1 : Y^2 + Y = X^3 + X + b$ where $b \in \{0, 1\}$ is a supersingular elliptic curve over $\mathbb{F}_q$ with $\#E_1(\mathbb{F}_q) = q \pm \sqrt{2q} + 1$. This elliptic curve has embedding degree 4, i.e., the smallest positive integer k for which $\#E_1(\mathbb{F}_q)$ divides $q^k - 1$ is $k = 4$. Then G is the order-n subgroup of $\mathbb{F}_{q^4}^*$. For example, if a 128-bit security level is desired, then one could use $E_1/\mathbb{F}_q : Y^2 + Y = X^3 + X$ with $q = 2^{1223}$ [1]. This curve has the property that $\#E_1(\mathbb{F}_q) = 5n$ where n is a 1221-bit prime, and so Pollard's rho method for solving the discrete logarithm problem in G_1 or G has running time approximately 2^{611}. Moreover, Coppersmith's algorithm [7] for solving the discrete logarithm problem in G has running time very roughly 2^{128} (see Table 6 of [16]).

Symmetric bilinear pairings can also be realized by selecting G_1 to be a group of points in $E_2(\mathbb{F}_q)$, where $q = 3^m$ and $E_2 : Y^2 = X^3 - X \pm 1$ is a supersingular elliptic curve over $\mathbb{F}_q$ with $\#E_2(\mathbb{F}_q) = q \pm \sqrt{3q} + 1$. This elliptic curve has embedding degree 6, and G is the order-n subgroup of $\mathbb{F}_{q^6}^*$. For example, if a 128-bit security level is desired, then one could use $E_2/\mathbb{F}_q : Y^2 = X^3 - X + 1$ with $q = 3^{509}$ [1]. This curve has the property that $\#E_2(\mathbb{F}_q) = 7n$ where n is a 804-bit prime, and so Pollard's rho method for solving the discrete logarithm problem in G_1 or G has running time approximately 2^{402}. Moreover, Coppersmith's algorithm for solving the discrete logarithm problem in G has running time very roughly 2^{128}.

In the Waters signature scheme [32], party A has a private key $Z = zP$ and a public key $\zeta = e(P, P)^z$. In order to sign a message M, A first computes $H = \text{Hash}(M) \in G_1$, where Hash is a cryptographic hash function that hashes its input elements into elliptic curve group elements, and chooses a random integer $r \in [1, n - 1]$. Then A computes $\alpha = Z + rH$ and $\beta = rP$ and sends (α, β) as her signature on M. A party B accepts A's signature on M if and only if $e(\alpha, P) = \zeta \cdot e(\beta, H)$. Our compression technique reduces the size of A's public key ζ by a factor of 4 or 6. More precisely, at the 128-bit security level, the size of the public key is reduced from 4892 bits to 1223 bits if E_1 is used, or from 4841 bits to 807 bits if E_2 is used. In order to verify A's signature, B can check if the compressed value of $e(\alpha, P)e(\beta, -H)$ is equal to the compressed value of ζ.

Another pairing-based application is the identity-based key agreement protocol of Scott [23]. In Scott's protocol, party A first computes a particular pairing value $g \in G$, and sends $P_A = g^a$ to party B. Similarly, B computes the pairing value $g \in G$ and sends $P_B = g^b$ to A. Finally, both A and B compute the shared secret $P_B^a = P_A^b$. (For details of the computations, please refer to [23].) If the symmetric bilinear pairings described above are employed, then messages exchanged can be compressed by factors of 4 or 6, and moreover the computations can take place over smaller fields rather than $\mathbb{F}_{q^4}$ or $\mathbb{F}_{q^6}$.

3. Preliminaries and notation

Let q be a prime power, and let $\mathbb{F}_q$ denote a finite field with q elements. Let n be a prime such that $\gcd(n, q) = 1$, and let k be the smallest positive integer such that $q^k \equiv 1 \pmod{n}$. Then $\mathbb{F}_{q^k}$ has a multiplicative subgroup of order n which cannot be embedded in the multiplicative group of any extension field $\mathbb{F}_{q^i}$ for $1 \leq i < k$. For such a triple (q, k, n) we denote the multiplicative group of order n by μ_n and call k the *embedding degree* of μ_n over $\mathbb{F}_q$.

Let $g \in \mathbb{F}_{q^k}$ and let s be a positive divisor of k. We assume that g is not contained in any proper subfield of $\mathbb{F}_{q^k}$. The *conjugates* of g over $\mathbb{F}_{q^s}$ are $g_i = g^{q^{is}}$ for $0 \leq i < k/s$. The *trace* of g over $\mathbb{F}_{q^s}$ is the sum of the conjugates of g over $\mathbb{F}_{q^s}$, i.e.,

$$\mathrm{Tr}_s(g) = \sum_{i=0}^{\frac{k}{s}-1} g_i \in \mathbb{F}_{q^s}. \tag{3.1}$$

The *minimal polynomial* of g over $\mathbb{F}_{q^s}$ is the monic polynomial

$$f_{g,s}(x) = \prod_{i=0}^{\frac{k}{s}-1} (x - g_i). \tag{3.2}$$

Note that $f_{g,s}(x) \in \mathbb{F}_{q^s}[x]$. When $s = 1$ we simply use $\mathrm{Tr}(g)$ and $f_g(x)$ by abuse of notation. Also, we will assume that the conjugates of g over $\mathbb{F}_{q^s}$ are well defined for any integer i by setting $g_i = g_{i \bmod k/s}$.

We fix some notation for finite field operations that will be used in the remainder of the paper. We will denote by $A_i, a_i, C_i, F_i, I_i, S_i, M_i$ and m_i the operations of addition, addition by 1 or 2, cubing, exponentiation by a power of the characteristic of the field, inversion, squaring, multiplication, and multiplication by 2 in $\mathbb{F}_{q^i}$ for $i = 1, 2, 3$. $SR_{i,j}$ will denote the cost of finding a root of a degree i irreducible polynomial over $\mathbb{F}_{q^j}$. We use *soft-O* notation $\tilde{O}(\cdot)$ as follows: $a = \tilde{O}(b)$ if and only if for some constant c, $a = O(b(\log_2 b)^c)$.

4. Multiplicative groups with embedding degree $k = 4$

In this section we concentrate on multiplicative groups μ_n with embedding degree $k = 4$ over $\mathbb{F}_q$. In other words, we fix parameters (q, n) such that q is a prime power, n is a prime, $\gcd(q, n) = 1$, $q^4 \equiv 1 \pmod{n}$, and $q^i \not\equiv 1 \pmod{n}$ for $1 \leq i < 4$. Finally, we let h be a positive integer and define $t_h = q + 1 - h \cdot n$ to be the *trace* of μ_n over $\mathbb{F}_q$ with respect to the *cofactor* h. Throughout the rest of this section we will assume that the cofactor h is fixed, and we simply denote the trace of μ_n by t instead of t_h.

In the following lemma we show that $g \in \mu_n$ together with its conjugates can be uniquely represented by the pair $(\mathrm{Tr}(g), \mathrm{Tr}(g^t))$ of $\mathbb{F}_q$-elements. This already gives us compression by a factor 2. Furthermore, we will show in Corollary 4.5 that in characteristic two finite fields it is possible to write all the coefficients of the minimal polynomial of g over $\mathbb{F}_q$ in terms of $\mathrm{Tr}(g)$ alone and hence achieve compression by a factor 4.

Lemma 4.1. *Let μ_n be the multiplicative subgroup of $\mathbb{F}^*_{q^4}$ of order n with embedding degree 4, trace t, and cofactor h. Let $g \in \mu_n$ and let g_i, for $i = 0, 1, 2, 3$, be the conjugates of g over $\mathbb{F}_q$. (Recall the convention that $g_i = g_{i \bmod 4}$.) Then*
(i) $g_i g_{i+1} = g_i^t$ for $i = 0, 1, 2, 3$.
(ii) $g_i g_{i+2} = 1$ for $i = 0, 1$.
(iii) $f_g(x) = x^4 - Tr(g)x^3 + (Tr(g^t) + 2)x^2 - Tr(g)x + 1$.

Proof. (i) $g_i g_{i+1} = g_i^{q+1} = g_i^t$ since g_i is of order n and $q + 1 - t \equiv 0 \pmod{n}$.
(ii) $g_i g_{i+2} = g_i^{q^2+1} = 1$ since g_i is of order n and $q^2 + 1 \equiv 0 \pmod{n}$.
(iii) Using (i) and (ii) gives

$$\begin{aligned} f_g(x) &= \prod_{i=0}^{3}(x - g_i) \\ &= x^4 - \left(\sum_{i=0}^{3} g_i\right) x^3 + \left(\sum_{0\le i<j\le 3} g_i g_j\right) x^2 - \left(\sum_{0\le i<j<k\le 3} g_i g_j g_k\right) x + 1 \\ &= x^4 - \mathrm{Tr}(g)x^3 + (\mathrm{Tr}(g^t) + 2)x^2 - \mathrm{Tr}(g)x + 1. \quad \square \end{aligned}$$

Suppose we fix a generator $g \in \mathbb{F}_{q^4}$ of μ_n, that is $\mu_n = \langle g \rangle$. In order to simplify the notation we define $c_u = \mathrm{Tr}(g^u)$ for any integer u. Note that $c_0 = 0$, $c_1 = \mathrm{Tr}(g)$, $c_u = c_{u \bmod n}$ and $c_{qu} = c_u^q = c_u$.

Lemma 4.2. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}^*_{q^4}$ of order n with embedding degree $k = 4$ and trace t. Then for all integers u and v we have*
(i) $c_u = c_{-u}$.
(ii) $c_u c_v = c_{u+v} + c_{u-v} + c_{u+v(t-1)} + c_{v+u(t-1)}$.

Proof. (i) This follows from Lemma 4.1(ii).
(ii) By Lemma 4.1(i) and (ii) it follows that

$$\begin{aligned} c_u c_v &= (g_0^u + g_1^u + g_2^u + g_3^u)(g_0^v + g_1^v + g_2^v + g_3^v) \\ &= \sum_{i=0}^{3} g_i^{u+v} + \sum_{i=0}^{3} g_i^u g_{i+1}^v + \sum_{i=0}^{3} g_i^u g_{i+2}^v + \sum_{i=0}^{3} g_i^u g_{i+3}^v \\ &= c_{u+v} + \sum_{i=0}^{3} g_i^{u-v}(g_i^v g_{i+1}^v) + \sum_{i=0}^{3} g_i^{u-v}(g_i^v g_{i+2}^v) + \sum_{i=0}^{3} g_i^v g_{i+1}^u \\ &= c_{u+v} + \sum_{i=0}^{3} g_i^{u+v(t-1)} + \sum_{i=0}^{3} g_i^{u-v} + \sum_{i=0}^{3} g_i^{v+u(t-1)} \\ &= c_{u+v} + c_{u+v(t-1)} + c_{u-v} + c_{v+u(t-1)}. \quad \square \end{aligned}$$

Theorem 4.3. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ of order n with embedding degree 4 and trace t. Then for all integers u and v we have*

$$c_{u+v} = c_u c_v - c_{u-v}(c_{tv} + 2) + c_{u-2v} c_v - c_{u-3v}. \tag{4.1}$$

Proof. First start with

$$\begin{aligned} c_u c_v &= (g_0^u + g_1^u + g_2^u + g_3^u)(g_0^v + g_1^v + g_2^v + g_3^v) \\ &= c_{u+v} + \sum_{i=0}^{3} \sum_{\substack{j=0 \\ j \neq i}}^{3} g_i^u g_j^v \\ &= c_{u+v} + \sum_{i=0}^{3} \sum_{\substack{j=0 \\ j \neq i}}^{3} g_i^{u-v}(g_i^v g_j^v) \\ &= c_{u+v} + \sum_{i=0}^{3} g_i^{u-v}(g_i^{tv} + g_{i+3}^{tv} + 1), \text{ by Lemma 4.1(i) and (ii)} \end{aligned}$$

$$= c_{u+v} + c_{u-v} + c_{u-v} c_{tv} - \sum_{i=0}^{3} g_i^{u-v}(g_{i+1}^{tv} + g_{i+2}^{tv}). \tag{4.2}$$

Then observe that

$$\sum_{i=0}^{3} g_i^{u-v} g_{i+1}^{tv} = \sum_{i=0}^{3} g_i^{u-v} g_i^{qtv}$$

$$= \sum_{i=0}^{3} g_i^{(u-2v)+qv} \text{ since } qt \equiv q^2 + q \equiv q - 1 \pmod{n} \tag{4.3}$$

and

$$\sum_{i=0}^{3} g_i^{u-v} g_{i+2}^{tv} = \sum_{i=0}^{3} g_i^{u-v} g_i^{-tv}$$

$$= \sum_{i=0}^{3} g_i^{(u-2v)-qv}, \text{ since } t \equiv q + 1 \pmod{n}. \tag{4.4}$$

Substituting (4.3) and (4.4) into (4.2) we obtain

$$c_u c_v = c_{u+v} + c_{u-v} + c_{u-v}c_{tv} - (c_{(u-2v)+qv} + c_{(u-2v)-qv}). \tag{4.5}$$

Now, let $u' = u - 2v, v' = qv$. Then

$$\begin{aligned} u' + v'(t-1) &\equiv (u-2v) + q^2 v \equiv u - 3v \pmod{n}, \\ v' + u'(t-1) &\equiv qv + (u-2v)q \equiv q(u-v) \pmod{n}, \end{aligned}$$

and using Lemma 4.2 (ii) with u', v' gives

$$\begin{aligned} c_{(u-2v)+qv} + c_{(u-2v)-qv} &= c_{u-2v}c_{qv} - (c_{u-3v} + c_{q(u-v)}) \\ &= c_{u-2v}c_v - (c_{u-3v} + c_{u-v}). \end{aligned} \tag{4.6}$$

Finally, (4.5) and (4.6) complete the proof. □

4.1. **Characteristic two finite fields.** Let r be a positive integer, and let $q = 2^{2r+1}$, $t = \pm 2^{r+1}$, $T = |t|$. The values of r for which $q + 1 - t = hn$ and n is prime lead to a multiplicative subgroup μ_n of $\mathbb{F}_{q^4}^*$ of prime order n with embedding degree 4. Throughout this section we fix h, n, q, t, T and $\mu_n = \langle g \rangle$ in this way, and also write $c_u = \mathrm{Tr}(g^u)$.

The following recursive relations follow from Theorem 4.3 by noting that the characteristic of $\mathbb{F}_q$ is 2 and also that $c_{vt} = c_{vT} = c_v^T$ (see Lemma 4.2(i)).

Corollary 4.4. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ with embedding degree 4 and trace t. Then for all integers u and v we have*
(i) $c_{u+v} = c_u c_v + c_{u-v}c_v^T + c_{u-2v}c_v + c_{u-3v}$.
(ii) $c_{2u} = c_u^2$.

Corollary 4.5. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ with embedding degree 4 and trace t. Let $f_{g^u}(x)$ be the minimal polynomial of $g^u \in \mu_n$ over $\mathbb{F}_q$. Then*

$$f_{g^u}(x) = x^4 + c_u x^3 + c_u^T x^2 + c_u x + 1.$$

Proof. The proof follows from Lemma 4.1(iii) and Corollary 4.4(ii). □

Remark 4.6. Throughout the remainder of this section we will assume without loss of generality that the trace t is positive. If t is negative then one can replace the expressions of the form $c_u^{p(t)}$, where p is some polynomial, by $c_u^{p(T)}$ without changing the validity of the results in this section.

4.2. **An exponentiation algorithm in μ_n.** Corollary 4.5 shows that the element $g^u \in \mu_n$ can be represented uniquely (up to conjugation) by its trace c_u. Our next objective is to develop an efficient method for computing c_a given c_1 and a; this is the exponentiation operation in μ_n. We define $s_1 = [c_{-1}, c_0, c_1, c_2] = [c_1, 0, c_1, c_1^2]$ to be the initial state. For a given state $s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}]$ with $u \geq 1$, if we can efficiently compute the states $s_{2u} = [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}]$ and $s_{2u+1} = [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}]$ then we immediately have an efficient double-and-add algorithm for computing c_a given c_1 and a.

Theorem 4.7. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ with embedding degree 4 and trace t. Let $c_u = Tr(g^u)$,*

$$A = \begin{pmatrix} c_1 & c_1 & 0 & 0 \\ 0 & c_1 & c_1 & 0 \\ 0 & 1 & c_1^t & 1 \\ 1 & c_1^t & 1 & 0 \end{pmatrix}, \ X = \begin{pmatrix} c_{2u-3} \\ c_{2u-1} \\ c_{2u+1} \\ c_{2u+3} \end{pmatrix}, \ Y = \begin{pmatrix} (c_u + c_{u-2})^2 + c_{u-1}^2 c_1^t \\ (c_{u+1} + c_{u-1})^2 + c_u^2 c_1^t \\ (c_u + c_{u+1})^2 c_1 \\ (c_{u-1} + c_u)^2 c_1 \end{pmatrix}.$$

Then
(i) A is invertible and $AX = Y$.
(ii) If c_1 is given then A and A^{-1} can be efficiently computed.
(iii) $c_{2u-1} = \frac{1}{c_1^{t+1}} \left((c_{u+1} + c_u + c_{u-1} + c_{u-2})^2 + (c_u + c_{u-1})^2 (c_1^t + c_1^2) \right)$.
(iv) $c_{2u+1} = c_{2u-1} + \frac{1}{c_1} \left((c_{u+1} + c_{u-1})^2 + c_u^2 c_1^t \right)$.

Proof. (i) Noting that the characteristic of $\mathbb{F}_q$ is 2, we can show that the determinant of A is equal to c_1^{t+2}. Hence A is invertible if and only if $c_1 \neq 0$. In fact, c_1 is never zero as otherwise the minimal polynomial $f_g(x) = x^4 + 1 = (x+1)^4$ is not irreducible. This proves the first part. For the second part we combine the four equations obtained from Corollary 4.4 with the following (u, v) values: $(2u-3, -1), (2u-2, -1), (2u-1, -1), (2u, -1)$, and also note that $c_{2u} = c_u^2$.
(ii) Let $A^{-1}[i]$ be the ith row of A^{-1}. Then one can check that

$$\begin{aligned} A^{-1}[1] &= [(c_1^t + 1)/c_1^{t+1}, 1/c_1^{t+1}, 0, 1/c_1^t], \\ A^{-1}[2] &= [1/c_1^{t+1}, 1/c_1^{t+1}, 0, 1/c_1^t], \\ A^{-1}[3] &= [1/c_1^{t+1}, (c_1^t + 1)/c_1^{t+1}, 0, 1/c_1^t], \\ A^{-1}[4] &= [(c_1^t + 1)/c_1^{t+1}, (c_1^{2t} + c_1^t + 1)/c_1^{t+1}, 1, (c_1^{t+1} + 1)/c_1^t] \end{aligned}$$

and the proof follows.
(iii) The inner product of $A^{-1}[2]$ and Y is equal to c_{2u-1}, and by part (ii) we can write

$$A^{-1}[2]Y = \frac{1}{c_1^{t+1}} \left((c_{u+1} + c_u + c_{u-1} + c_{u-2})^2 + (c_u + c_{u-1})^2 (c_1^t + c_1^2) \right).$$

(iv) The proof is similar to the proof of part (iii). □

The formulas for c_{2u-1} and c_{2u+1} in Theorem 4.7 yield Algorithm 1 for exponentiation in μ_n.

Remark 4.8. Algorithm 1 can be used to compute c_{ab} given c_a and b as follows. We set $c'_1 = c_a$ and the initial state becomes $s'_1 = [c'_{-1}, c'_0, c'_1, c'_2] = [c_a, 0, c_a, c_a^2]$. With input c'_1 and b, Algorithm 1 outputs $c'_b = c_{ab}$.

Since the cost of addition is negligible in finite fields of characteristic two, we will ignore addition costs in the performance analysis of algorithms in this section. Moreover, we may ignore the cost $(1F_1 + 1S_1)$ in the precomputation steps of Algorithm 1 as it is negligible comparing to $(1I_1 + 1M_1)$. Then the cost of Algorithm 1 can be approximated as:

Precomputation (steps 2 and 3): $1I_1 + 1M_1$.

Main loop (steps 4–15): $(4M_1 + 4S_1)(\ell - 1)$.

Algorithm 1 Computing c_a
Input: c_1 and a
Output: c_a

```
Write $a = \sum_{i=0}^{\ell-1} a_i 2^i$ where $a_i \in \{0,1\}$ and $a_{\ell-1} = 1$
$s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}] \leftarrow [c_1, 0, c_1, c_1^2]$
$m_1 \leftarrow 1/c_1^{t+1}$ and $m_2 \leftarrow 1/c_1$
for $i$ from $\ell - 2$ down to $0$ do
    $c_{2u-1} \leftarrow m_1 \left((c_{u+1} + c_u + c_{u-1} + c_{u-2})^2 + (c_u + c_{u-1})^2 (c_1^t + c_1^2)\right)$
    $c_{2u} \leftarrow c_u^2$
    $c_{2u+1} \leftarrow c_{2u-1} + m_2 \left((c_{u+1} + c_{u-1})^2 + c_u^2 c_1^t\right)$
    if $a_i = 1$ then
        $c_{2u+2} \leftarrow c_{u+1}^2$
        $s_u \leftarrow [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}]$
    else
        $c_{2u-2} \leftarrow c_{u-1}^2$
        $s_u \leftarrow [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}]$
    end if
end for
Return $(c_u)$
```

We note that Algorithm 1 has a limited degree of built-in resistance to side-channel analysis attacks because the same types of operations are executed whether the bit a_i of the exponent is 1 or 0.

4.3. **Other algorithms for exponentiation with compressed elements.** Algorithm 1 works directly with the factor-4 compressed elements. In this section, we describe four algorithms for computing c_{ab} given c_a and b. The first algorithm partially decompresses c_a to an element $\tilde{c}_a \in \mathbb{F}_{q^2}$, and then uses the LUC method for exponentiating in this representation. The second algorithm decompresses c_a to an element in $\mathbb{F}_{q^4}$, and then employs a standard window-NAF exponentiation method. The third and fourth methods use the Brouwer-Pellikaan-Verheul idea (cf. Appendix A.2) by using minimal polynomials over $\mathbb{F}_{q^2}$ and $\mathbb{F}_q$, respectively. The five exponentiation algorithms are compared in Section 6.

First, we prove the following.

Lemma 4.9. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^4}^*$ with embedding degree 4, trace t, and cofactor h. Let $c_u = Tr(g^u)$ and $\tilde{c}_u = Tr_2(g^u)$. Then $\{\tilde{c}_u, \tilde{c}_u^q\}$ is the set of roots of the polynomial $\tilde{f}_{g^u}(x) = x^2 + c_u x + c_u^t$.*

Proof. Since g has order n and $q + 1 \equiv t \pmod{n}$ and $q^2 \equiv -1 \pmod{n}$, we have $(x - \tilde{c}_u)(x - \tilde{c}_u^q) = (x - (g^u + g^{uq^2}))(x - (g^{uq} + g^{uq^3})) = x^2 + c_u x + c_u^t = \tilde{f}_{g^u}(x)$. □

4.3.1. *An algorithm based on the LUC cryptosystem.* We first describe an algorithm to compute c_a given c_1 and a. The idea of the algorithm is as follows. Let $\tilde{d}_1 = \tilde{c}_1^q$. Suppose we know an element in the set $\{\tilde{c}_1, \tilde{d}_1\}$. If $\tilde{c}_1$ is known then we will compute $\tilde{c}_a$, and if $\tilde{d}_1$ is known then we will compute $\tilde{d}_a$. In both cases, we can determine $c_a = \tilde{c}_a + \tilde{c}_a^q = \tilde{d}_a + \tilde{d}_a^q$.

Now, by Lemma 4.9 one can determine $\{\tilde{c}_1, \tilde{d}_1\}$ from c_1 by finding the roots of the polynomial $\tilde{f}_g(x) = x^2 + c_1 x + c_1^t$ in $\mathbb{F}_{q^2}$. By the argument in the previous paragraph, we may assume without loss of generality that $\tilde{c}_1$ is known. Note that the minimal polynomial of g over $\mathbb{F}_{q^2}$ is $f_{g,2}(x) = x^2 + \tilde{c}_1 x + 1$, and for all integers u and v we have the following recursive relation (see [27] or Appendix A.1):

$$\tilde{c}_{u+v} = \tilde{c}_u \tilde{c}_v - \tilde{c}_{u-v}.$$

In particular, since the characteristic of the field is 2, we have $\tilde{c}_{2u} = \tilde{c}_u^2$ and $\tilde{c}_{2u+1} = \tilde{c}_{u+1}\tilde{c}_u + \tilde{c}_1$. Thus, if we define $s_u = [\tilde{c}_u, \tilde{c}_{u+1}]$, then $s_1 = [\tilde{c}_1, \tilde{c}_1^2]$, $s_{2u} = [\tilde{c}_u^2, \tilde{c}_{u+1}\tilde{c}_u + \tilde{c}_1]$, and $s_{2u+1} = [\tilde{c}_{u+1}\tilde{c}_u + \tilde{c}_1, \tilde{c}_{u+1}^2]$. This leads to the double-and-add algorithm described in Algorithm 2. We note that Algorithm 2 can be used to compute c_{ab} given c_a and b (cf. Remark 4.8).

Algorithm 2 Computing c_a based on the LUC cryptosystem
Input: c_1 and a
Output: c_a

```
1: Write $a = \sum_{i=0}^{\ell-1} a_i 2^i$ where $a_i \in \{0,1\}$ and $a_{\ell-1} = 1$
2: $\tilde{c}_1 \leftarrow$ a root of the polynomial $x^2 + c_1 x + c_1^t$, $\tilde{c}_1 \in \mathbb{F}_{q^2}$
3: $s_u = [\tilde{c}_u, \tilde{c}_{u+1}] \leftarrow [\tilde{c}_1, \tilde{c}_1^2]$
4: for $i$ from $\ell - 2$ down to $0$ do
5:     $\tilde{c}_{2u+1} \leftarrow \tilde{c}_{u+1}\tilde{c}_u + \tilde{c}_1$
6:     if $a_i = 1$ then
7:         $\tilde{c}_{2u+2} \leftarrow \tilde{c}_{u+1}^2$
8:         $s_u \leftarrow [\tilde{c}_{2u+1}, \tilde{c}_{2u+2}]$
9:     else
10:        $\tilde{c}_{2u} \leftarrow \tilde{c}_u^2$
11:        $s_u \leftarrow [\tilde{c}_{2u}, \tilde{c}_{2u+1}]$
12:    end if
13: end for
14: Return $(\tilde{c}_u + \tilde{c}_u^q)$
```

We may ignore the costs $(1F_1 + 1S_2)$ and $1F_2$ in the precomputation steps and in step 14 of Algorithm 2 as they are dominated by $SR_{2,1}$ and $(1M_2 + 1S_2)(\ell - 1)$, respectively. Then the cost of Algorithm 2 can be approximated as:

Precomputation (steps 2 and 3): $1SR_{2,1}$.

Main loop (steps 4–14): $(1M_2 + 1S_2)(\ell - 1)$.

4.3.2. *Decompressing and direct exponentiation in μ_n (Algorithm DDE).* Given c_a and an ℓ-bit integer b, in order to compute c_{ab} we will decompress c_a to g^a (or to one of its conjugates over $\mathbb{F}_q$). Then we will compute g^{ab} (up to conjugation over $\mathbb{F}_q$). Finally, summing the four conjugates of g^{ab} over $\mathbb{F}_q$ gives c_{ab}.

In order to decompress c_a we first construct the polynomial $\tilde{f}_{g^a}(x) = x^2 + c_a x + c_a^t$ (see Lemma 4.9) over $\mathbb{F}_q$ and find a root in $\mathbb{F}_{q^2}$; without loss of generality, suppose that this root is $\tilde{c}_a$. Next we construct the minimal polynomial of g^a over $\mathbb{F}_{q^2}$, i.e., $f_{g^a,2}(x) = x^2 + \tilde{c}_a x + 1$ and find a root of $f_{g^a,2}(x)$ in $\mathbb{F}_{q^4}$. Hence we obtain g^a

or one of its conjugates over $\mathbb{F}_q$. The decompression can be achieved at a cost of $1SR_{2,1} + 1SR_{2,2}$ (we ignore the cost $1F_1$ that is dominated by $SR_{2,1}$ and $SR_{2,2}$).

Now, to exponentiate $g^a \in \mu_n$ (or one of its conjugates over $\mathbb{F}_q$) to the power b, one first determines the *width-w NAF* representation of b, i.e., $b = \sum_{i=0}^{\ell'} b_i 2^i$ where $\ell' \in \{\ell - 1, \ell\}$, $b_{\ell'} \neq 0$, each nonzero b_i is odd, $|b_i| < 2^{w-1}$, and at most one of any w consecutive digits is nonzero. The width-w NAF representation of b contains on average $\ell/(w+1)$ nonzero digits (see [19] for properties of width-w NAF representations). After precomputing and storing elements $g_i = g^{ai}$ for $i \in \{\pm 1, \pm 3, \pm 5, \ldots, \pm 2^{w-1} - 1\}$, one can compute g^{ab} at an average cost of ℓ squarings and $\ell/(w+1)$ multiplications in $\mathbb{F}_{q^4}$. Finally, we note that Karatsuba's technique can be used to multiply two elements in $\mathbb{F}_{q^4}$ at a cost of $9M_1$, and squaring in μ_n can be performed at a cost of $4S_1$. Note that by choosing a suitable polynomial for the extension $\mathbb{F}_{q^4}/\mathbb{F}_q$, we may ignore the cost of polynomial reductions in the extension field arithmetic. We may also ignore the cost for computing g_i's in the precomputation step as it is dominated by $SR_{2,1}$ and $SR_{2,2}$ and the cost for computing the sum of the four conjugates of g^{ab}.

Hence, the expected cost of computing c_{ab} can be approximated as $1SR_{2,1} + 1SR_{2,2} + (4S_1 + \frac{9}{(w+1)}M_1)\ell$.

4.3.3. *Direct exponentiation in μ_n without decompressing (Algorithm BPV-I).* This algorithm is based on the idea of Brouwer, Pellikaan and Verheul (see [6] or Appendix A.2). Suppose c_a and an ℓ-bit integer b are given. By Lemma 4.9, we can determine the minimal polynomial of g^a or g^{aq} over $\mathbb{F}_{q^2}$ at a cost of $1SR_{2,1}$ (we ignore the cost $1F_1$ that is dominated by $SR_{2,1}$). Without loss of generality let's assume that we know $f_{g^a,2}(x) = x^2 + \tilde{c}_a x + 1$. That is, we have a copy of $\mathbb{F}_{q^4} = \mathbb{F}_{q^2}[x]/(f_{g^a,2}(x))$ and next we compute x^b modulo $f_{g^a,2}(x)$ using the conventional repeated square-and-multiply algorithm. Since $(\tau_1 x + \tau_0)^2 = (\tau_1^2 \tilde{c}_a)x + (\tau_0^2 + \tau_1^2)$, the squaring step can be achieved at a cost of $1M_2 + 2S_2$. And, since $(\tau_1 x + \tau_0)x = (\tau_1 \tilde{c}_a)x + (\tau_1 + \tau_2)$, the multiplication step can be achieved at a cost of $1M_2$. Therefore, computing $x^b = w_1 x + w_0$ with $w_i \in \mathbb{F}_{q^2}$ costs on average $((1M_2 + 2S_2) + \frac{1}{2}M_2)(\ell - 1)$. Finally, we compute $c_{ab} = \mathrm{Tr}(x^b) = \mathrm{Tr}(\mathrm{Tr}_2(w_1 x) + \mathrm{Tr}_2(w_0)) = \mathrm{Tr}(w_1 \tilde{c}_a) = w_1 \tilde{c}_a + (w_1 \tilde{c}_a)^q$ at a cost of $1F_2 + 1M_2$. Hence, the approximate expected cost of the algorithm is $1SR_{2,1} + ((1M_2 + 2S_2) + \frac{1}{2}M_2)(\ell - 1)$ (we ignore the cost $(1F_2 + 1M_2)$ in the last step that is dominated by the cost of the main loop).

4.3.4. *Direct exponentiation in μ_n without decompressing (Algorithm BPV-II).* The idea of the algorithm is similar to Algorithm BPV-I, except that we work with a minimal polynomial over $\mathbb{F}_q$ instead of $\mathbb{F}_{q^2}$. Given c_a and b, we first determine $f_{g^a}(x) = x^4 + c_a x^3 + c_a^t x^2 + c_a x + 1$ at a cost of $1F_1$. Now, we have a copy of $\mathbb{F}_{q^4} = \mathbb{F}_q[x]/(f_{g^a}(x))$ and next we compute x^b modulo $f_{g^a}(x)$ using the conventional repeated square-and-multiply algorithm. Since $x^6 = (c_a^3 + c_a)x^3 + (c_a^{2t} + c_a^{t+2} + c_a^2 + 1)x^2 + (c_a^{t+1} + c_a^3 + c_a)x + (c_a^t + c_a^2)$ modulo $f_{g^a}(x)$, and $(\tau_3 x^3 + \tau_2 x^2 + \tau_1 x + \tau_0)^2 = \tau_3^2 x^6 + \tau_2^2 x^4 + \tau_1^2 x^2 + \tau_0^2$, the squaring step can be achieved at a cost of $6M_1 + 4S_1$. And, since $(\tau_3 x^3 + \tau_2 x^2 + \tau_1 x + \tau_0)x = (\tau_3 c_a + \tau_2)x^3 + (\tau_3 c_a^t + \tau_1)x^2 + (\tau_3 c_a + \tau_0)x + \tau_3$, the multiplication step can be achieved at a cost of $2M_1$. Therefore, computing $x^b = w_3 x^3 + w_2 x^2 + w_1 x + w_0$ with $w_i \in \mathbb{F}_q$ costs $((6M_1 + 4S_1) + 1M_1)(\ell - 1)$. Finally, using $x^4 = c_a x^3 + c_a^t x^2 + c_a x + 1$ in $\mathbb{F}_{q^4} = \mathbb{F}_q[x]/(f_{g^a}(x))$, we can compute

$c_{ab} = \mathrm{Tr}(x^b) = \mathrm{Tr}(w_3x^3 + w_2x^2 + w_1x + w_0) = w_3c_{3a} + w_2c_{2a} + w_1c_a = w_3c_a(c_a^2 + c_a^t + 1) + w_2c_a^2 + w_1c_a$ at a cost of $1F_1 + 4M_1 + 1S_1$. Hence, the approximate expected cost of the algorithm is $1F_1 + (7M_1 + 4S_1)(\ell - 1)$ (we ignore the cost $(1F_1 + 4M_1 + 1S_1)$ in the last step that is dominated by the cost of the main loop).

5. Multiplicative groups with embedding degree $k = 6$

In this section we concentrate on multiplicative groups μ_n with embedding degree $k = 6$ over $\mathbb{F}_q$. In other words, we fix parameters (q, n) such that q is a prime power, n is a prime, $\gcd(q, n) = 1$, $q^6 \equiv 1 \pmod{n}$, and $q^i \not\equiv 1 \pmod{n}$ for $1 \le i < 6$. Finally, we let h be a positive integer and define $t_h = q + 1 - h \cdot n$ to be the *trace* of μ_n over $\mathbb{F}_q$ with respect to the *cofactor* h. Throughout the rest of this section we will assume that the cofactor h is fixed, and we simply denote the trace of μ_n by t instead of t_h.

In the following lemma we show that $g \in \mu_n$ together with its conjugates can be uniquely represented by the triple ($\mathrm{Tr}(g)$, $\mathrm{Tr}(g^t)$, $\mathrm{Tr}(g^2)$) of $\mathbb{F}_q$-elements. In fact, it is easy to show that $\mathrm{Tr}(g^2)$ can be written in terms of $\mathrm{Tr}(g)$ and $\mathrm{Tr}(g^t)$. This already gives us compression by a factor of 3 . Furthermore, as first proven by Shirase et al. [25], Corollary 5.6 shows that in characteristic three finite fields it is possible to write all the coefficients of the minimal polynomial of g over $\mathbb{F}_q$ in terms of $\mathrm{Tr}(g)$ alone and hence achieve compression by a factor of 6.

Lemma 5.1. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ of order n with embedding degree 6, trace t, and cofactor h. Let $g \in \mu_n$ and let g_i, for $i = 0, 1, \ldots, 5$, be the conjugates of g over $\mathbb{F}_q$. (Recall the convention that $g_i = g_{i \bmod 6}$.) Then*
(i) $g_ig_{i+1} = g_i^t$ for $i = 0, 1, \ldots, 5$.
(ii) $g_ig_{i+2} = g_{i+1}$ for $i = 0, 1, \ldots, 5$.
(iii) $g_ig_{i+3} = 1$ for $i = 0, 1, 2$.
(iv) $f_g(x) = x^6 - Tr(g)x^5 + (Tr(g^t) + Tr(g) + 3)x^4 - (Tr(g^2) + 2Tr(g) + 2)x^3 + (Tr(g^t) + Tr(g) + 3)x^2 - Tr(g)x + 1$.

Proof. (i) $g_ig_{i+1} = g_i^{q+1} = g_i^t$ since g_i is of order n and $q + 1 - t \equiv 0 \pmod{n}$.
(ii) $g_ig_{i+2} = g_i^{q^2+1} = g_{i+1}$ since g_i is of order n and $q^2 + 1 \equiv q \pmod{n}$.
(iii) $g_ig_{i+3} = g_i^{q^3+1} = 1$ since g_i is of order n and $q^3 + 1 \equiv 0 \pmod{n}$.
(iv) By (iii) we can write

$$\begin{aligned} f_g(x) &= \prod_{i=0}^{5}(x - g_0) \\ &= x^6 - \left(\sum_{i=0}^{5} g_i\right)x^5 + \left(\sum_{0\le i<j\le 5} g_ig_j\right)x^4 - \left(\sum_{0\le i<j<k\le 5} g_ig_jg_k\right)x^3 \\ &\quad + \left(\sum_{0\le i<j\le 5} g_ig_j\right)x^2 - \left(\sum_{i=0}^{5} g_i\right)x + 1. \end{aligned}$$

Moreover, by (i), (ii), and (iii) we have

$$\begin{aligned}\sum_{0\le i<j\le 5} g_i g_j &= \sum_{i=0}^{5} g_i g_{i+1} + \sum_{i=0}^{5} g_i g_{i+2} + \sum_{i=0}^{2} g_i g_{i+3}\\ &= \sum_{i=0}^{5} g_i^t + \sum_{i=0}^{5} g_{i+1} + \sum_{i=0}^{2} 1\\ &= \mathrm{Tr}(g^t) + \mathrm{Tr}(g) + 3\end{aligned}$$

and

$$\begin{aligned}\sum_{0\le i<j<k\le 5} g_i g_j g_k &= \sum_{i=0}^{5} g_i g_{i+1} g_{i+2} + \sum_{i=0}^{2} \sum_{\substack{j=0\\ j\not\equiv i,i+3 \pmod 6}}^{5} g_i g_{i+3} g_j + \sum_{i=0}^{1} g_i g_{i+2} g_{i+4}\\ &= \sum_{i=0}^{5} g_{i+1}^2 + 2\sum_{j=0}^{5} g_j + 2\end{aligned}$$

which completes the proof. □

Remark 5.2. Barreto and Naehrig [3, Section 3] suggested that pairing values for the asymmetric pairing derived from Barreto-Naehrig (BN) elliptic curves can be compressed to one-sixth of their length. BN pairing values lie in the subgroup $\mu_n \subset \mathbb{F}_{p^{12}}^*$ where $n = n(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$, $p = p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$, and $x \in \mathbb{Z}$ is such that $n(x)$ and $p(x)$ are prime. Their compression method identifies the elements of the subgroup μ_n of $\mathbb{F}_{q^6}^*$ (where $q = p^2$) with their traces over $\mathbb{F}_q$. In fact, given $q = p^2$ and n as above, one can write $t = q + 1 - h \cdot n$ where t is the trace of the corresponding BN curve over $\mathbb{F}_q$, $p \nmid t$, and show that μ_n has embedding degree 6 over $\mathbb{F}_q$ with trace t. However, as one can see from Lemma 5.1, $\mathrm{Tr}(g)$ does not suffice to identify an element $g \in \mu_n$ uniquely up to its conjugates over $\mathbb{F}_q$. Hence, it is possible that there are *collisions* for the trace function. That is, there may exist elements $h_1, h_2 \in \mu_n \subset \mathbb{F}_{q^6}^*$ such that h_1 and h_2 are not conjugates of each other over $\mathbb{F}_q$ and $\mathrm{Tr}(h_1) = \mathrm{Tr}(h_2)$, in which case the Barreto-Naehrig compression method fails. We searched for collisions in the case $x \in \{-41, -15, -7, -3, -2, -1, 1, 5, 6, 7, 20\}$ and discovered one when $x = 6$ (where $n = 55117$ and $q = (55333)^2$). In Appendix B we list one BN and eight BN-like sets of parameters (p, n, T, g, u, v) such that $n \mid (p^4 - p^2 + 1)$, $\mu_n = \langle g\rangle$ is the order-n subgroup of $\mathbb{F}_{q^6}^*$, and (g^u, g^v) is a collision with colliding value $T = \mathrm{Tr}(g^u) = \mathrm{Tr}(g^v)$.

Suppose we fix a generator $g \in \mathbb{F}_{q^6}$ of μ_n, that is $\mu_n = \langle g\rangle$. In order to simplify the notation we define $c_u = \mathrm{Tr}(g^u)$ for any integer u. Note that $c_0 = 0$, $c_1 = \mathrm{Tr}(g)$, $c_u = c_{u \bmod n}$ and $c_{qu} = c_u^q = c_u$.

Lemma 5.3. *Let $\mu_n = \langle g\rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ of order n with embedding degree 6 and trace t. Then for all integers u and v we have*
(i) $c_u = c_{-u}$.
(ii) $c_u c_v = c_{u+v} + c_{u-v} + c_{u+v(t-1)} + c_{v+u(t-1)} + c_{u+v(t-2)} + c_{v+u(t-2)}$.

Proof. (i) This follows from Lemma 5.1 (iii).
(ii) First note that

$$\begin{aligned} c_u c_v &= (g_0^u + g_1^u + g_2^u + g_3^u + g_4^u + g_5^u)(g_0^v + g_1^v + g_2^v + g_3^v + g_4^v + g_5^v) \\ &= \sum_{i=0}^{5} g_i^{u+v} + \sum_{i=0}^{5} g_i^u g_{i+1}^v + \sum_{i=0}^{5} g_i^u g_{i+2}^v \\ &\quad + \sum_{i=0}^{5} g_i^u g_{i+3}^v + \sum_{i=0}^{5} g_i^u g_{i+4}^v + \sum_{i=0}^{5} g_i^u g_{i+5}^v. \end{aligned}$$

Observing the equations below with the help of Lemma 5.1(i), (ii) and (iii)

$$\begin{aligned} \sum_{i=0}^{5} g_i^u g_{i+1}^v &= \sum_{i=0}^{5} g_i^{u-v}(g_i^v g_{i+1}^v) = \sum_{i=0}^{5} g_i^{u+v(t-1)}, \\ \sum_{i=0}^{5} g_i^u g_{i+2}^v &= \sum_{i=0}^{5} g_i^{u-v}(g_i^v g_{i+2}^v) = \sum_{i=0}^{5} g_i^{u-v} g_{i+1}^v = \sum_{i=0}^{5} g_i^{u+v(t-2)}, \end{aligned}$$

$$\begin{aligned} \sum_{i=0}^{5} g_i^u g_{i+3}^v &= \sum_{i=0}^{5} g_i^{u-v}(g_i^v g_{i+3}^v) = \sum_{i=0}^{5} g_i^{u-v}, \\ \sum_{i=0}^{5} g_i^u g_{i+4}^v &= \sum_{i=0}^{5} g_i^v g_{i+2}^u = \sum_{i=0}^{5} g_i^{v+u(t-2)}, \\ \sum_{i=0}^{5} g_i^u g_{i+5}^v &= \sum_{i=0}^{5} g_i^v g_{i+1}^u = \sum_{i=0}^{5} g_i^{v+u(t-1)} \end{aligned}$$

leads us to the result. □

Theorem 5.4. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ of order n with embedding degree 6 and trace t. Then for all integers u and v we have*

$$c_{u+v} = (c_u + c_{u-4v})c_v - (c_{u-v} + c_{u-3v})(c_{tv} + c_v + 3) + c_{u-2v}(2c_v + c_{2v} + 2) - c_{u-5v}.$$

Proof. First we start with

$$\begin{aligned}
c_u c_v &= (g_0^u + g_1^u + g_2^u + g_3^u + g_4^u + g_5^u)(g_0^v + g_1^v + g_2^v + g_3^v + g_4^v + g_5^v) \\
&= \sum_{i=0}^{5} g_i^{u+v} + \sum_{i=0}^{5} \sum_{\substack{j=0 \\ j \neq i}}^{5} g_i^u g_j^v \\
&= c_{u+v} + \sum_{i=0}^{5} \sum_{\substack{j=0 \\ j \neq i}}^{5} g_i^{u-v} (g_i^v g_j^v) \\
&= c_{u+v} + \sum_{i=0}^{5} g_i^{u-v} (g_i^{tv} + g_{i+5}^{tv} + g_{i+1}^{v} + g_{i+5}^{v} + 1) \\
&= c_{u+v} + c_{u-v} + c_{u-v} c_{tv} + c_{u-v} c_v \\
&\quad - \left(\sum_{i=0}^{5} g_i^{u-v} (g_{i+1}^{tv} + g_{i+2}^{tv} + g_{i+3}^{tv} + g_{i+4}^{tv}) \right) \\
&\quad - \left(\sum_{i=0}^{5} g_i^{u-v} (g_i^{v} + g_{i+2}^{v} + g_{i+3}^{v} + g_{i+4}^{v}) \right).
\end{aligned}$$

Now, since

$$\begin{aligned}
\sum_{i=0}^{5} g_i^{u-v} g_{i+1}^{tv} &= \sum_{i=0}^{5} g_i^{u-v+qtv}, \text{ by Lemma 5.1(i)} \\
&= c_{q^4(u-v)+q^5tv}, \text{ since } c_{qu} = c_u \\
&= c_{qu-2v}, \text{ by Lemma 5.3(i) and } q^4 \equiv -q \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+2}^{tv} &= \sum_{i=0}^{5} g_i^{u-v+q^2tv}, \text{ by Lemma 5.1(i)} \\
&= c_{(u-3v)+qv} \text{ since } q^2 t \equiv q - 2 \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+3}^{tv} &= \sum_{i=0}^{5} g_i^{u-v+q^3tv}, \text{ by Lemma 5.1(i)} \\
&= c_{(u-2v)-qv} \text{ since } q^3 \equiv -1 \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+4}^{tv} &= \sum_{i=0}^{5} g_i^{u-v+q^4tv}, \text{ by Lemma 5.1(i)} \\
&= c_{u-2qv}, \text{ by Lemma 5.3(i) and } q^4 t \equiv -2q + 1 \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+2}^{v} &= \sum_{i=0}^{5} g_i^{u-v+q^2v}, \text{ by Lemma 5.1(i)} \\
&= c_{(u-2v)+qv} \text{ since } q^2 \equiv q - 1 \pmod{n}, \\
\sum_{i=0}^{5} g_i^{u-v} g_{i+3}^{v} &= \sum_{i=0}^{5} g_i^{u-v+q^3v}, \text{ by Lemma 5.1(i)} \\
&= c_{(u-2v)} \text{ since } q^3 \equiv -1 \pmod{n},
\end{aligned}$$

and

$$\begin{aligned}\sum_{i=0}^{5} g_i^{u-v} g_{i+4}^{v} &= \sum_{i=0}^{5} g_i^{u-v+q^4 v}, \text{ by Lemma 5.1(i)}\\ &= c_{(u-v)-qv} \text{ since } q^4 \equiv -q \pmod{n},\end{aligned}$$

we have

$$\begin{aligned}c_u c_v &= c_{u+v} + c_{u-v} + c_{u-v}c_{tv} + c_{u-v}c_v\\ &\quad -(c_{qu-2v} + c_{(u-3v)+qv} + c_{(u-2v)-qv} + c_{u-2qv})\\ &\quad -(c_u + c_{(u-2v)+qv} + c_{u-2v} + c_{(u-v)-qv}).\end{aligned} \tag{5.1}$$

In order to eliminate some of the terms in equation 5.1 let us use Lemma 5.3(ii) with $u' = u - 2v$, and $v' = qv$. By noting that

$$\begin{aligned}c_{v'} &= c_v, \text{ since } c_{qu} = c_u,\\ c_{u'+v'} &= c_{(u-2v)+qv},\\ c_{u'-v'} &= c_{(u-2v)-qv},\\ c_{u'+v'(t-1)} &= c_{(u-3v)+qv}, \text{ since } q(t-1) \equiv q^2 \equiv q-1 \pmod{n},\\ c_{u'+v'(t-2)} &= c_{(u-3v)},\\ c_{v'+u'(t-1)} &= c_{u-v}, \text{ since } t-1 \equiv q \pmod{n} \text{ and } c_{qu} = c_u,\\ c_{v'+u'(t-2)} &= c_{q(q(u-v)-(u-2v))}, \text{ since } c_{qu} = c_u\\ &= c_{(u-v)-qv}, \text{by Lemma 5.3(i)},\end{aligned}$$

we can rewrite (5.1) as

$$\begin{aligned}c_u c_v &= c_{u+v} + 2c_{u-v} + c_{u-v}c_{tv} + c_{u-v}c_v\\ &\quad -(c_{qu-2v} + c_{u-2qv} + c_u + c_{u-2v})\\ &\quad -(c_{u-2v}c_v - c_{u-3v}).\end{aligned} \tag{5.2}$$

By replacing v by $-v$ in (5.2) we obtain another equation and summing this with (5.2) gives

$$\begin{aligned}c_{u+3v} &= 2c_u c_v - 3(c_{u+v} + c_{u-v}) - (c_{tv} + c_v)(c_{u+v} + c_{u-v}) + 2c_u\\ &\quad + c_v(c_{u+2v} + c_{u-2v}) + (c_{u+2v} + c_{u-2v}) - c_{u-3v} + c_{qu-2v}\\ &\quad + c_{u-2qv} + c_{qu+2v} + c_{u+2qv}.\end{aligned} \tag{5.3}$$

Also, using Lemma 5.3(ii) with $u' = u$ and $v' = 2v$, and noting that

$$\begin{aligned}c_{u'+v'(t-1)} &= c_{u+2qv}, \text{ since } t-1 \equiv q \pmod{n},\\ c_{u'+v'(t-2)} &= c_{q(u+2(q-1)v)}, \text{ since } c_{qu} = c_u \text{ and } t-2 \equiv q-1 \pmod{n},\\ &= c_{qu-2v}, \text{ since } q^2 \equiv q-1 \pmod{n},\\ c_{v'+u'(t-1)} &= c_{qu+2v}, \text{ since } t-1 \equiv q \pmod{n},\\ c_{v'+u'(t-2)} &= c_{q(2v+(q-1)u)}, \text{ since } c_{qu} = c_u \text{ and } t-2 \equiv q-1 \pmod{n},\\ &= c_{-u+2qv} = c_{u-2qv}, \text{ since } q^2 \equiv q-1 \pmod{n} \text{ and } c_u = c_{-u},\end{aligned}$$

we can rewrite (5.3) as

$$\begin{aligned}c_{u+3v} &= 2c_u c_v - 3(c_{u+v} + c_{u-v}) - (c_{tv} + c_v)(c_{u+v} + c_{u-v}) + 2c_u\\ &\quad + c_v(c_{u+2v} + c_{u-2v}) - c_{u-3v} + c_u c_{2v}.\end{aligned} \tag{5.4}$$

Finally, using (5.4) with $u' = u - 2v$ and $v' = v$ results in

$$\begin{aligned} c_{u+v} &= c_u c_v - (c_{u-v} + c_{u-3v})(c_{tv} + c_v + 3) + c_{u-2v}(2c_v + c_{2v} + 2) \\ &\quad + c_{u-4v}c_v - c_{u-5v}, \end{aligned}$$

as required. □

5.1. **Characteristic three finite fields.** This section follows along the same lines as Section 4.1. Let r be a positive integer, and let $q = 3^{2r+1}$, $t = \pm 3^{r+1}, T = |t|$. The values of r for which $q + 1 - t = hn$ and n is prime lead to a multiplicative subgroup μ_n of $\mathbb{F}_{q^6}^*$ of prime order n with embedding degree 6. Throughout this section we fix h, n, q, t, T and $\mu_n = \langle g \rangle$ in this way, and also write $c_u = \mathrm{Tr}(g^u)$.

The following recursive relations follow from Theorem 5.4, using Lemma 5.3(ii) with $u = v$, and by noting that the characteristic of $\mathbb{F}_q$ is 3 and also that $c_{vt} = c_{vT} = c_v^T$ (see Lemma 5.3(i)).

Corollary 5.5. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ with embedding degree 6 and trace t. Then for all integers u and v we have*
(i) $c_{2u} = c_u^2 + c_u + c_u^T$.
(ii) $c_{u+v} = c_u c_v - (c_{u-v} + c_{u-3v})(c_v^T + c_v) + c_{u-2v}(c_v^2 + c_v^T + 2) + c_{u-4v}c_v - c_{u-5v}$.

Corollary 5.6. *Let μ_n be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ with embedding degree 6 and trace t. Let $f_{g^u}(x)$ be the minimal polynomial of $g^u \in \mu_n$ over $\mathbb{F}_q$. Then*

$$f_{g^u}(x) = x^6 - c_u x^5 + (c_u^T + c_u)x^4 - (c_u^2 + c_u^T + 2)x^3 + (c_u^T + c_u)x^2 - c_u x + 1.$$

Proof. The proof follows from Lemma 5.1(iv), Corollary 5.5(i) and from the fact that $c_{vt} = c_{vT} = c_v^T$. □

Remark 5.7. Throughout the remainder of this section we will assume without loss of generality that the trace t is positive. If t is negative then one can replace the expressions of the form $c_u^{p(t)}$, where p is some polynomial, by $c_u^{p(T)}$ without changing the validity of the results in this section.

5.2. **An exponentiation algorithm in μ_n.** Corollary 5.6 shows that the element g^u can be represented uniquely (up to conjugation) by its trace c_u. Similarly as in Section 5.2, our aim is to develop an efficient method to compute c_a given c_1 and a. We define $s_1 = [c_{-1}, c_0, c_1, c_2, c_3, c_4]$ to be the initial state. For a given state $s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}, c_{u+2}, c_{u+3}]$ with $u \geq 1$, if we can efficiently compute the states $s_{2u} = [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}, c_{2u+3}]$ and $s_{2u+1} = [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}, c_{2u+3}, c_{2u+4}]$ then we immediately have an efficient double-and-add algorithm for computing c_a given c_1 and a.

Theorem 5.8. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ with embedding degree 6 and trace t. Let c_u = $Tr(g^u)$, $C_{1,t} = c_1 + c_1^t$, $C_{1,t,2} = c_1^2 + c_1^t + 2$,*

$M = (c_1^t(c_1^3 + c_1^{t+1} + 2(c_1^t + 1)) + 2(c_1^3 + c_1 + 1))^{-1}$, and

$$A = \begin{pmatrix} 1 & C_{1,t} & C_{1,t} & 1 & 0 & 0 \\ 0 & 1 & C_{1,t} & C_{1,t} & 1 & 0 \\ 0 & 0 & 1 & C_{1,t} & C_{1,t} & 1 \\ c_1 & C_{1,t,2} & c_1 & 0 & 0 & 0 \\ 0 & c_1 & C_{1,t,2} & c_1 & 0 & 0 \\ 0 & 0 & c_1 & C_{1,t,2} & c_1 & 0 \end{pmatrix}, \; X = \begin{pmatrix} c_{2u-3} \\ c_{2u-1} \\ c_{2u+1} \\ c_{2u+3} \\ c_{2u+5} \\ c_{2u+7} \end{pmatrix},$$

$$Y = \begin{pmatrix} (c_{2u+2} + c_{2u-2})c_1 + c_{2u}C_{1,t,2} \\ (c_{2u+4} + c_{2u})c_1 + c_{2u+2}C_{1,t,2} \\ (c_{2u+6} + c_{2u+2})c_1 + c_{2u+4}C_{1,t,2} \\ (c_{2u} + c_{2u-2})C_{1,t} + c_{2u+2} + c_{2u-4} \\ (c_{2u+2} + c_{2u})C_{1,t} + c_{2u+4} + c_{2u-2} \\ (c_{2u+4} + c_{2u+2})C_{1,t} + c_{2u+6} + c_{2u} \end{pmatrix}.$$

Then
(i) A is invertible and $AX = Y$.
(ii) If c_1 is given then A and A^{-1} can be efficiently computed.
(iii) $c_{2u-1} = M(c_1^3+2c_1^2+2c_1+c_1^{t+1}, c_1^2, 0, 2(c_1^2+c_1^t)+c_1+1, 2(c_1^2+c_1^{t+1})+c_1, 2c_1)Y$.
(iv) $c_{2u+1} = M(2c_1^2, 2c_1^2, 0, c_1, 2c_1^t + c_1^{t+1} + c_1 + 1, c_1)Y$.
(v) $c_{2u+3} = M(c_1^2, c_1^3+2c_1^2+2c_1+c_1^{t+1}, 0, 2c_1, 2(c_1^2+c_1^{t+1})+c_1, 2(c_1^2+c_1^t)+c_1+1)Y$.

Proof. (i) Noting that the characteristic of $\mathbb{F}_q$ is 3 and using Corollary 5.5(i), we can show that determinant of A is equal to $(c_1-1)^{2t+2}(c_2-1)$. Hence A is invertible if and only if $c_1 \neq 1$ and $c_2 \neq 1$. In fact we can show that $c_1 \neq 1$ and $c_2 \neq 1$ as follows. If $c_1 = 1$ then $f_g(x) = x^6 - x^5 + 2x^4 - x^3 + 2x^2 - x + 1 = (x^2+x+1)(x^2-x+1)^2$, and if $c_2 = 1$ then $f_g(x) = (x^2+1)(x^4 - c_1x^3 + (c_1 + c_1^t - 1)x^2 - c_1x + 1)$. Both cases contradict the irreducibility of the minimal polynomial $f_g(x)$ over $\mathbb{F}_q$. This proves that A is invertible. Now, combining the six equations obtained from Corollary 5.5 with the following (u, v) values: $(2u-3, -1), (2u-2, -1), (2u-1, -1), (2u, -1), (2u+1, -1), (2u+2, -1)$ and using Corollary 5.5(i) proves that $AX = Y$.
(ii) The proof follows from part (i) and from the definition of A.
(iii) A computation in Maple shows that the second row of A^{-1} is $A^{-1}[2] = \frac{1}{M}(c_1^3 + 2c_1^2 + 2c_1 + c_1^{t+1}, c_1^2, 0, 2(c_1^2 + c_1^t) + c_1 + 1, 2(c_1^2 + c_1^{t+1}) + c_1), 2c_1)$. The rest follows as $c_{2u-1} = A^{-1}[2]Y$ from part (i).
(iv) We compute $A^{-1}[3] = \frac{1}{M}(2c_1^2, 2c_1^2, 0, c_1, 2c_1^t + c_1^{t+1} + c_1 + 1, c_1)$ and conclude from $c_{2u+1} = A^{-1}[3]Y$.
(v) We compute $A^{-1}[4] = \frac{1}{M}(c_1^2, c_1^3+2c_1^2+2c_1+c_1^{t+1}, 0, 2c_1, 2(c_1^2+c_1^{t+1})+c_1), 2(c_1^2 + c_1^t) + c_1 + 1)$ and conclude from $c_{2u+3} = A^{-1}[4]Y$. □

The formulas for $c_{2u-1}, c_{2u+1}, c_{2u+3}$ in Theorem 5.8 yield Algorithm 3 for exponentiation in μ_n.

Remark 5.9. Algorithm 3 can be used to compute c_{ab} given c_a and b as follows. We replace $c'_1 = c_a$ and the initial state becomes $s'_1 = [c'_{-1}, c'_0, c'_1, c'_2, c'_3, c'_4] = [c_a, 0, c_a, C_{a,t,2}, c_a^3, C_{a,t,2}^2 + C_{a,t,2} + C_{a,t,2}^t]$. With input c'_1 and b, Algorithm 3 outputs $c'_b = c_{ab}$.

We may ignore the costs $(9A_1 + 4a_1 + 1C_1 + 1F_1 + 2m_1)$ in the precomputation steps of Algorithm 3 and $(3a_1 + 10m_1)(\ell - 1)$ in the main loop of Algorithm 3 as

Algorithm 3 Computing c_a
Input: c_1 and a
Output: c_a

```
1: Write $a = \sum_{i=0}^{\ell-1} a_i 2^i$ where $a_i \in \{0,1\}$ and $a_{\ell-1} = 1$
2: $C_2 \leftarrow c_1^2, \quad C_t \leftarrow c_1^{t+1}$
3: $C_{1,t} \leftarrow c_1 + c_1^t, \quad C_{1,t,2} \leftarrow C_2 + c_1^t + 2$
4: $c_2 \leftarrow C_{1,t,2} + c_1 + 1, \quad c_4 \leftarrow c_2^2 + c_2 + c_2^t$
5: $s_u = [c_{u-2}, c_{u-1}, c_u, c_{u+1}, c_{u+2}, c_{u+3}] \leftarrow [c_1, 0, c_1, c_2, c_1^3, c_4]$
6: $M \leftarrow (c_1^t(c_1^3 + C_t + 2(c_1^t + 1)) + 2(c_1^3 + c_1 + 1))^{-1}$
7: **for** $i$ **from** $\ell - 2$ **down to** $0$ **do**
8:     $c_{2u-4} \leftarrow c_{u-2}^2 + c_{u-2} + c_{u-2}^t, \quad c_{2u-2} \leftarrow c_{u-1}^2 + c_{u-1} + c_{u-1}^t,$
       $c_{2u} \leftarrow c_u^2 + c_u + c_u^t, \quad c_{2u+2} \leftarrow c_{u+1}^2 + c_{u+1} + c_{u+1}^t,$
       $c_{2u+4} \leftarrow c_{u+2}^2 + c_{u+2} + c_{u+2}^t, \quad c_{2u+6} \leftarrow c_{u+3}^2 + c_{u+3} + c_{u+3}^t$
9:     $Y_1 \leftarrow (c_{2u+2} + c_{2u-2})c_1 + c_{2u}C_{1,t,2},$
       $Y_2 \leftarrow (c_{2u+4} + c_{2u})c_1 + c_{2u+2}C_{1,t,2},$
       $Y_4 \leftarrow (c_{2u} + c_{2u-2})C_{1,t} + c_{2u+2} + c_{2u-4},$
       $Y_5 \leftarrow (c_{2u+2} + c_{2u})C_{1,t} + c_{2u+4} + c_{2u-2},$
       $Y_6 \leftarrow (c_{2u+4} + c_{2u+2})C_{1,t} + c_{2u+6} + c_{2u}$
10:    $c_{2u-1} \leftarrow M((c_1^3 + 2(C_2 + c_1) + C_t)Y_1 + C_2Y_2 + (2(C_2 + c_1^t) + c_1 + 1)Y_4$
              $+(2(C_2 + C_t) + c_1)Y_5 + 2c_1Y_6)$
11:    $c_{2u+1} \leftarrow M(2C_2(Y_1 + Y_2) + c_1(Y_4 + Y_6) + (2c_1^t + C_t + c_1 + 1)Y_5)$
12:    $c_{2u+3} \leftarrow M(C_2Y_1 + (c_1^3 + 2(C_2 + c_1) + C_t)Y_2 + 2c_1Y_4$
              $+(2(C_2 + C_t) + c_1)Y_5 + (2(C_2 + c_1^t) + c_1 + 1)Y_6)$
13:    **if** $a_i = 1$ **then**
14:        $s_u \leftarrow [c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}, c_{2u+3}, c_{2u+4}]$
15:    **else**
16:        $s_u \leftarrow [c_{2u-2}, c_{2u-1}, c_{2u}, c_{2u+1}, c_{2u+2}, c_{2u+3}]$
17:    **end if**
18: **end for**
19: Return $(c_u)$
```

they are dominated by $(1I_1 + 2M_1 + 2S_1)$ and $(53A_1 + 6F_1 + 23M_1 + 6S_1)(\ell - 1)$, respectively. Then the cost of Algorithm 3 can be approximated as:

Precomputation (steps 2–6): $1I_1 + 2M_1 + 2S_1$.

Main loop (steps 7–18): $(53A_1 + 6F_1 + 23M_1 + 6S_1)(\ell - 1)$.

We note that Algorithm 3 has a limited degree of built-in resistance to side-channel analysis attacks as the same types of operations are executed whether the bit a_i of the exponent is 1 or 0.

5.3. **Other algorithms for exponentiation with compressed elements.** Algorithm 3 works directly with the factor-6 compressed elements. In this section, we describe five algorithms for computing c_{ab} given c_a and b. The first algorithm partially decompresses c_a to an element $\tilde{c}_a \in \mathbb{F}_{q^3}$, and then uses the LUC method for exponentiating in this representation. The second algorithm decompresses c_a to an element in $\mathbb{F}_{q^6}$, and then employs a standard window-NAF exponentiation method. The third, fourth and fifth methods use the Brouwer-Pellikaan-Verheul idea (cf. Appendix A.2) by using minimal polynomials over $\mathbb{F}_{q^2}$, $\mathbb{F}_{q^3}$ and $\mathbb{F}_q$, respectively. The seven exponentiation algorithms including XTR$_3$ [25] are compared in Section 6.

First, we prove the following.

Lemma 5.10. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}^*_{q^6}$ with embedding degree* 6 *and trace t. Also, let $c_u = Tr(g^u)$ and $\tilde{c}_u = Tr_3(g^u)$. Then $\{\tilde{c}_u, \tilde{c}_u^q, \tilde{c}_u^{q^2}\}$ is the set of roots of the polynomial $\tilde{f}_{g^u}(x) = x^3 - c_u x^2 + (c_u + c_u^t)x - (c_u^2 + c_u^t - 2c_u + 2)$.*

Proof. Let $\tilde{d}_u = \tilde{c}_u^q$ and $\tilde{e}_u = \tilde{c}_u^{q^2}$. Since g has order n and $q + 1 \equiv t \pmod{n}$ and $q^3 \equiv -1 \pmod{n}$, we can rewrite the minimal polynomial of g^u over $\mathbb{F}_q$ as

$$\begin{aligned}
f_{g^u}(x) &= \Big[(x - g^u)(x - g^{q^3 u})\Big]\Big[(x - g^{qu})(x - g^{q^4 u})\Big]\Big[(x - g^{q^2 u})(x - g^{q^5 u})\Big] \\
&= (x^2 - \tilde{c}_u x + 1)(x^2 - \tilde{d}_u x + 1)(x^2 - \tilde{e}_u x + 1) \\
&= x^6 - c_u x^5 + (\tilde{c}_u \tilde{d}_u + \tilde{d}_u \tilde{e}_u + \tilde{c}_u \tilde{e}_u)x^4 - (2c_u + \tilde{c}_u \tilde{d}_u \tilde{e}_u)x^3 \\
&\quad + (\tilde{c}_u \tilde{d}_u + \tilde{d}_u \tilde{e}_u + \tilde{c}_u \tilde{e}_u)x^2 - c_u x + 1.
\end{aligned}$$

Comparing the coefficients of this polynomial with the coefficients of $f_{g^u}(x)$ as given in Corollary 5.6 we get

$$\begin{aligned}
\tilde{c}_u \tilde{d}_u + \tilde{d}_u \tilde{e}_u + \tilde{c}_u \tilde{e}_u &= c_u + c_u^t \\
\tilde{c}_u \tilde{d}_u \tilde{e}_u &= c_u^2 + c_u^t - 2c_u + 2.
\end{aligned}$$

The proof then follows because

$$\begin{aligned}
(x - \tilde{c}_u)(x - \tilde{d}_u)(x - \tilde{e}_u) &= x^3 - (\tilde{c}_u + \tilde{d}_u + \tilde{e}_u)x^2 + (\tilde{c}_u \tilde{d}_u + \tilde{d}_u \tilde{e}_u + \tilde{c}_u \tilde{e}_u)x \\
&\quad - \tilde{c}_u \tilde{d}_u \tilde{e}_u \\
&= x^3 - c_u x^2 + (c_u + c_u^t)x - (c_u^2 + c_u^t - 2c_u + 2) \\
&= \tilde{f}_{g^u}(x). \quad \square
\end{aligned}$$

We will also need the following lemma which was proven in [25].

Lemma 5.11. *Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}^*_{q^6}$ with embedding degree* 6 *and trace t. Also, let $c_u = Tr(g^u)$ and $\tilde{c}_u = Tr_2(g^u)$. Then $\{\tilde{c}_u, \tilde{c}_u^q\}$ is the set of roots of the polynomial $\tilde{f}_{g^u}(x) = x^2 - c_u x + c_u^t$.*

5.3.1. *An algorithm based on the LUC cryptosystem.* We will describe a ternary exponentiation algorithm to compute c_a given c_1 and a. The idea of the algorithm is as follows. Suppose we know an element in the set $\{\tilde{c}_1, \tilde{c}_1^q, \tilde{c}_1^{q^2}\}$. If $\tilde{c}_1$ is known then we will compute $\tilde{c}_a$, if $\tilde{c}_1^q$ is known then we will compute $\tilde{c}_a^q$, and if $\tilde{c}_1^{q^2}$ is known then we will compute $\tilde{c}_a^{q^2}$. In all three cases, we can determine $c_a = \tilde{c}_a + \tilde{c}_a^q + \tilde{c}_a^{q^2} = \tilde{c}_a^q + (\tilde{c}_a^q)^q + (\tilde{c}_a^q)^{q^2} = \tilde{c}_a^{q^2} + (\tilde{c}_a^{q^2})^q + (\tilde{c}_a^{q^2})^{q^2}$.

By Lemma 5.10 we can determine $\{\tilde{c}_1, \tilde{c}_1^q, \tilde{c}_1^{q^2}\}$ based on c_1 by finding the roots of the polynomial $\tilde{f}_g(x) = x^3 - c_1 x^2 + (c_1 + c_1^t)x - (c_1^2 + c_1^t - 2c_1 + 2)$. From our argument in the previous paragraph, we may assume without loss of generality that $\tilde{c}_1$ is known.

Note that the minimal polynomial of g over $\mathbb{F}_{q^3}$ is $f_{g,3}(x) = x^2 - \tilde{c}_1 x + 1$, and for all integers u and v we have the following recursive relation (see [27] or Appendix A.1):

$$\tilde{c}_{u+v} = \tilde{c}_u \tilde{c}_v - \tilde{c}_{u-v}. \tag{5.5}$$

Moreover, it was shown in [24] that the following equations can be deduced from (5.5).

$$\begin{aligned}
\tilde{c}_{3u} &= \tilde{c}_u^3 && (5.6)\\
\tilde{c}_{3u-1} &= \frac{1}{\tilde{c}_1^2 - 1}(\tilde{c}_{u-1}^3 + \tilde{c}_1 \tilde{c}_u^3) && (5.7)\\
\tilde{c}_{3u-1} &= \frac{1}{\tilde{c}_1^2 - 1}((\tilde{c}_1^3 + \tilde{c}_1)\tilde{c}_u^3 - \tilde{c}_{u+1}^3) && (5.8)\\
\tilde{c}_{3u+1} &= \frac{1}{\tilde{c}_1^2 - 1}((\tilde{c}_1^3 + \tilde{c}_1)\tilde{c}_u^3 - \tilde{c}_{u-1}^3) && (5.9)\\
\tilde{c}_{3u+1} &= \frac{1}{\tilde{c}_1^2 - 1}(\tilde{c}_{u+1}^3 + \tilde{c}_1 \tilde{c}_u^3). && (5.10)
\end{aligned}$$

We now describe how to obtain $\tilde{c}_a$ given $\tilde{c}_1$ and a; to the author's knowledge this exponentiation method was first presented in [24] to compute Lucas sequences. We will assume that a is written in signed ternary notation, i.e., $a = \sum_{i=0}^{\ell-1} a_i 3^i$ where $a_i \in \{-1, 0, 1\}$ and $a_{\ell-1} = 1$ First, define two states: $s_u^{(0)} = [\tilde{c}_u, \tilde{c}_{u+1}]$ and $s_u^{(1)} = [\tilde{c}_{u-1}, \tilde{c}_u]$. The algorithm begins with the state $s_1^{(1)} = [\tilde{c}_0, \tilde{c}_1] = [2, \tilde{c}_1]$, and exactly one of the two states, $s_u^{(j)}, j = 0, 1$, will be active during the execution of the algorithm.

If $j = 0$ in the ith step of the algorithm, then $s_u^{(0)}$ is active and we will compute $\tilde{c}_{3u}$ based on (5.6) and compute one of $\tilde{c}_{3u-1}$ or $\tilde{c}_{3u+1}$ based on (5.8) or (5.10). In this case, if $a_i = 0$ then we set $j = 0$ (i.e., $s_u^{(0)}$ remains active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$; if $a_i = 1$ we set $j = 1$ (i.e., $s_u^{(1)}$ becomes active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$; and if $a_i = -1$ we set $j = 0$ (i.e., $s_u^{(0)}$ remains active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$.

If $j = 1$ in the ith step of the algorithm, then $s_u^{(1)}$ is active and we will compute $\tilde{c}_{3u}$ based on (5.6) and compute one of $\tilde{c}_{3u-1}$ or $\tilde{c}_{3u+1}$ based on (5.7) or (5.9). In this case, if $a_i = 0$ then we set $j = 1$ (i.e., $s_u^{(1)}$ remains active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$; if $a_i = 1$ we set $j = 1$ (i.e., $s_u^{(1)}$ remains active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$; and if $a_i = -1$ we set $j = 0$ (i.e., $s_u^{(0)}$ becomes active) and $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$.

At the end of this procedure we obtain $\tilde{c}_a$ from one of the active states $s_a^{(0)} = [\tilde{c}_a, \tilde{c}_{a+1}]$ or $s_a^{(1)} = [\tilde{c}_{a-1}, \tilde{c}_a]$, as required. The exponentiation method is summarized in Algorithm 4. We may ignore the costs $(2A_1 + 2a_1 + 1A_3 + 1C_3 + 1F_1 + 1I_1 + 2m_1 + 1S_1)$ in the precomputation steps of Algorithm 4 and $(1m_3)(\ell - 1) + (2A_3 + 2F_3)$ in the main loop of Algorithm 4 as they are dominated by $1SR_{3,1}$ and $(2C_3 + 2M_3)(\ell - 1)$, respectively. Then the cost of Algorithm 4 can be approximated as:

Precomputation (steps 2–4): $1SR_{3,1}$.

Main loop (steps 5–37): $(1A_3 + 2C_3 + 2M_3)(\ell - 1)$.

Algorithm 4 Computing c_a based on the LUC cryptosystem
Input: c_1 and a
Output: c_a

```
Write $a = \sum_{i=0}^{\ell-1} a_i 3^i$ where $a_i \in \{-1, 0, 1\}$ and $a_{\ell-1} = 1$
$C \leftarrow 1/(c_1^2 - 1)$
$\tilde{c}_1 \leftarrow$ a root of the polynomial $x^3 - c_1 x^2 + (c_1 + c_1^t)x - (c_1^2 + c_1^t + c_1 + 2)$
$\tilde{C} \leftarrow \tilde{c}_1^3 + \tilde{c}_1$
$j \leftarrow 1$
$s_u^{(j)} = [\tilde{c}_{u-1}, \tilde{c}_u] \leftarrow [2, \tilde{c}_1]$
for $i$ from $\ell - 2$ down to $0$ do
    $\tilde{c}_{3u} \leftarrow \tilde{c}_u^3$
    if $j = 0$ then
        $\{s_u^{(0)} = [\tilde{c}_u, \tilde{c}_{u+1}]$ is active$\}$
        if $a_i = -1$ then
            $\tilde{c}_{3u-1} \leftarrow C(\tilde{C}\tilde{c}_u^3 - \tilde{c}_{u+1}^3)$ {see (5.8)}
            $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$
        else if $a_i = 0$ then
            $\tilde{c}_{3u+1} \leftarrow C(\tilde{c}_{u+1}^3 + \tilde{c}_1\tilde{c}_u^3)$ {see (5.10)}
            $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$
        else if $a_i = 1$ then
            $\tilde{c}_{3u+1} \leftarrow C(\tilde{c}_{u+1}^3 + \tilde{c}_1\tilde{c}_u^3)$ {see (5.10)}
            $j \leftarrow 1$
            $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$
        end if
    else
        $\{s_u^{(1)} = [\tilde{c}_{u-1}, \tilde{c}_u]$ is active$\}$
        if $a_i = -1$ then
            $\tilde{c}_{3u-1} \leftarrow C(\tilde{c}_{u-1}^3 + \tilde{c}_1\tilde{c}_u^3)$ {see (5.7)}
            $j \leftarrow 0$
            $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$
        else if $a_i = 0$ then
            $\tilde{c}_{3u-1} \leftarrow C(\tilde{c}_{u-1}^3 + \tilde{c}_1\tilde{c}_u^3)$ {see (5.7)}
            $s_u^{(j)} \leftarrow [\tilde{c}_{3u-1}, \tilde{c}_{3u}]$
        else if $a_i = 1$ then
            $\tilde{c}_{3u+1} \leftarrow C(\tilde{C}\tilde{c}_u^3 - \tilde{c}_{u-1}^3)$ {( see 5.9)}
            $s_u^{(j)} \leftarrow [\tilde{c}_{3u}, \tilde{c}_{3u+1}]$
        end if
    end if
end for
Return $(\tilde{c}_u + \tilde{c}_u^q + \tilde{c}_u^{q^2})$
```

Remark 5.12. Montgomery [18] presented several methods (binary, binary-ternary, CFRC, PRAC) to compute Lucas sequences. The main idea in his methods to compute $\tilde{c}_a$ is to build a Lucas chain for a and at each step in the chain to use the recursive formula $\tilde{c}_{u+v} = \tilde{c}_u\tilde{c}_v - \tilde{c}_{u-v}$ for some suitable u and v (see Table 4 in [18]). The length of the derived Lucas chains in these algorithms exceed $1.446(\log_2 a)$ (see

Theorem 8 and Table 5 in [18]) and each step in the chain requires at least $1M_3$ (see Table 4 in [18]). Therefore, these methods seem unlikely to outperform Algorithm 4 whose cost might be approximated as $2(\log_3 2)(\log_2 a)M_3 \approx 1.26(\log_2 a)M_3$ after the precomputation step.

5.3.2. *Decompressing and direct exponentiation in μ_n (Algorithm DDE).* Given c_a and an integer b, in order to compute c_{ab} we will first decompress c_a to g^a or one of its conjugates over $\mathbb{F}_q$. Then we will compute g^{ab} (up to conjugation over $\mathbb{F}_q$) by working directly in $\mathbb{F}_{q^6}$. Finally, summing the six conjugates of g^{ab} gives c_{ab}.

In order to decompress g^a we first construct the polynomial $\tilde{f}_{g^a}(x) = x^3 - c_a x^2 + (c_a + c_a^t)x - (c_a^2 + c_a^t - 2c_a + 2)$ (see Lemma 5.10) over $\mathbb{F}_q$ and find a root in $\mathbb{F}_{q^3}$; without loss of generality, suppose that this root is $\tilde{c}_a$. Next we construct the minimal polynomial of g^a over $\mathbb{F}_{q^3}$, i.e., $f_{g^a,3}(x) = x^2 - \tilde{c}_a x + 1$ and find a root of $f_{g^a,3}(x)$ in $\mathbb{F}_{q^6}$. Hence we obtain g^a or one of its conjugates over $\mathbb{F}_q$. The decompression can be achieved at a cost of $4A_1 + 1a_1 + 1F_1 + 1S_1 + 2m_1 + 1SR_{3,1} + 1SR_{2,3}$.

Now, to exponentiate $g^a \in \mu_n$ (or one of its conjugates over $\mathbb{F}_q$) to the power b, one first determines the *width-w radix-3 NAF* representation of b, i.e., $b = \sum_{i=0}^{\ell} b_i 3^i$ where $b_\ell > 0$, each nonzero b_i is nonzero modulo 3, and $|b_i| \leq (3^w - 1)/2$. The width-w radix-3 NAF representation of b contains on average $2\ell/(2w+1)$ nonzero digits (see [29] for more details on width-w radix-3 NAF representations). After precomputing and storing some elements one can compute g^{ab} at an average cost of l cubings and $2l/(2w+1)$ multiplications in $\mathbb{F}_{q^6}$. Using Karatsuba's technique, multiplying two elements in $\mathbb{F}_{q^6}$ can be accomplished with 18 multiplications in $\mathbb{F}_q$. Cubing in μ_n can be performed at a cost of $6C_1$. Note that by choosing a suitable polynomial for the extension $\mathbb{F}_{q^6}/\mathbb{F}_q$, we may ignore the costs of polynomial reductions in the extension field arithmetic.

Hence, the average cost of computing c_{ab} can be approximated as $1SR_{3,1} + 1SR_{2,3} + (6C_1 + \frac{36}{(2w+1)}M_1)\ell$.

5.3.3. *Direct exponentiation in μ_n without decompressing (Algorithm BPV-I).* This algorithm is based on the idea of Brouwer, Pellikaan and Verheul (see [6] or Appendix A.2). Suppose c_a and an integer $b = \sum_{i=0}^{\ell} b_i 3^i$, where $b_i \in \{0, 1, 2\}$, in base-3 representation is given. By Lemma 5.11, we can determine $\{\tilde{c}_a, \tilde{d}_a\}$, the set of traces of g^a and g^{aq} over $\mathbb{F}_{q^2}$. Then the set of minimal polynomials of g^a and g^{aq} over $\mathbb{F}_{q^2}$ are $f_{g^a,2}(x) = x^3 - \tilde{c}_a x^2 + \tilde{c}_a^q x - 1$ and $f_{g^{aq},2}(x) = x^3 - \tilde{d}_a x^2 + \tilde{d}_a^q x - 1$. Without loss of generality let's assume that we know $f_{g^a,2}(x)$ (which can be computed at a cost of $1F_1 + 1F_2 + 1m_1 + 1SR_{2,1}$). That is, we have a copy of $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[x]/(f_{g^a,2}(x))$ and next we compute x^b modulo $f_{g^a,2}(x)$ using the repeated cube-and-multiply algorithm. Since $(\tau_2 x^2 + \tau_1 x + \tau_0)^3 = \tau_2^3 x^6 + \tau_1^3 x^3 + \tau_0^3$, each cubing (modulo $f_{g^a,2}(x)$) can be achieved at a cost of $4A_2 + 3C_2 + 5M_2$. Now, since $(\tau_2 x^2 + \tau_1 x + \tau_0)x = (\tau_2 \tilde{c}_a + \tau_1)x^2 + (2\tau_2 \tilde{c}_a^q + \tau_0)x + \tau_2$, multiplying by x can be achieved at a cost of $2A_2 + 1m_2 + 2M_2$. Similarly, we can show that multiplying by x^2 can be achieved at a cost of $3A_2 + 2m_2 + 4M_2$. Therefore, computing $x^b = w_2 x^2 + w_1 x + w_0$ with $w_i \in \mathbb{F}_{q^2}$ costs on average $(4A_2 + 3C_2 + 5M_2 + \frac{1}{3}(5A_2 + 3m_2 + 6M_2))(\ell - 1)$. Now, $c_{ab} = \mathrm{Tr}(x^b) = \mathrm{Tr}(w_2 x^2 + w_1 x + w_0) = \mathrm{Tr}(w_2 \mathrm{Tr}_2(x^2) + w_1 \mathrm{Tr}_2(x))$. Also note that $\mathrm{Tr}_2(x) = \tilde{c}_a$, and $\mathrm{Tr}_2(x^2) = \tilde{c}_a^2 + \tilde{c}_a^q$ follows from $x^3 = \tilde{c}_a x^2 - \tilde{c}_a^q x + 1$ in $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}[x]/(f_{g^a,2}(x))$. Hence, we can write

$$c_{ab} = w_1\tilde{c}_a + (w_1\tilde{c}_a)^q + w_2(\tilde{c}_a^2 + \tilde{c}_a^q) + (w_2(\tilde{c}_a^2 + \tilde{c}_a^q))^q$$

and the expected cost of computing c_{ab} is $1SR_{2,1} + (\frac{17}{3}A_2 + 3C_2 + 7M_2))(\ell - 1)$ (we ignore the cost $(1F_1 + 1F_2 + 1m_1)$ that is dominated by $SR_{2,1}$ in the precomputation steps, and the costs $(3m_2)(\ell - 1)$ in the main loop and $(2F_2 + 2M_2)$ in the last step).

5.3.4. *Direct exponentiation in μ_n without decompressing (Algorithm BPV-II).* The idea of the algorithm is similar to Algorithm BPV-I except that we work with a minimal polynomial over $\mathbb{F}_{q^3}$ instead of $\mathbb{F}_{q^2}$. Suppose c_a and an integer $b = \sum_{i=0}^{\ell} b_i 3^i$ in width-w radix-3 representation is given. By Lemma 5.10, we can determine the set of minimal polynomials of g^{aq^i} for $i = 0, 1, 2$. Without loss of generality let's assume that we know $f_{g^a,3}(x) = x^2 - \tilde{c}_a x + 1$ (which can be computed at a cost of $4A_1 + 1a_1 + 1F_1 + 1S_1 + 2m_1 + 1SR_{3,1}$). That is, we have a copy of $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[x]/(f_{g^a,3}(x))$ and next we compute x^b modulo $f_{g^a,3}(x)$ using width-w radix-3 NAF exponentiation. Since $(\tau_1 x + \tau_0)^3 = \tau_1^3(\tilde{c}_a^2 - 1)x + (\tilde{c}_a\tau_1^3 + \tau_0^3)$ modulo $f_{g^a,3}(x)$, the cubing step can be achieved at a cost of $1A_1 + 1a_1 + 2C_3 + 2M_3 + 1m_3$. Using Karatsuba's technique, we can show that multiplying two elements in $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[x]/(f_{g^a,3}(x))$ can be accomplished at a cost of $4M_3$. Therefore, computing $x^b = w_1 x + w_0$, $w_i \in \mathbb{F}_{q^3}$, costs $(1A_1 + 1a_1 + 2C_3 + 2M_3 + 1m_3 + \frac{8}{(2w+1)}M_3)\ell$. Hence, the expected cost of computing $c_{ab} = \mathrm{Tr}(x^b) = \mathrm{Tr}(w_1 x + w_0) = \mathrm{Tr}(w_1\mathrm{Tr}_3(x) + \mathrm{Tr}_3(w_0)) = \mathrm{Tr}(w_1\tilde{c}_a + 2w_0) = w_1\tilde{c}_a + (w_1\tilde{c}_a)^q + (w_1\tilde{c}_a)^{q^2}$ is $1SR_{3,1} + (1A_1 + 2C_3 + 2M_3 + \frac{8}{(2w+1)}M_3)\ell$ (we ignore the cost $(4A_1 + 1a_1 + 1F_1 + 1S_1 + 2m_1)$ that is dominated by $SR_{3,1}$ in the precomputation steps, and the costs $(1a_1 + 1m_3)\ell$ in the main loop and $(2A_3 + 2F_3 + 1M_3)$ in the last step).

5.3.5. *Direct exponentiation in μ_n without decompressing (Algorithm BPV-III).* The idea of the algorithm is similar to Algorithm BPV-I and BPV-II except that we work with a minimal polynomial over $\mathbb{F}_q$ instead of $\mathbb{F}_{q^2}$ or $\mathbb{F}_{q^3}$. Given c_a and $b = \sum_{i=0}^{\ell} b_i 3^i$ in width-w radix-3 representation, we first determine $f_{g^a}(x) = x^6 - c_a x^5 + (c_a^t + c_a)x^4 - (c_a^2 + c_a^t + 2)x^3 + (c_a^t + c_a)x^2 - c_a x + 1$ at a cost of $1S_1$ (we ignore the cost $2A_1 + 1a_1 + 1F_1 + 2m_1$). Now, we have a copy of $\mathbb{F}_{q^6} = \mathbb{F}_q[x]/(f_{g^a}(x))$, and next we compute x^b modulo $f_{g^a}(x)$ using width-w radix-3 NAF exponentiation. Since $(\tau_5 x^5 + \tau_4 x^4 + \tau_3 x^3 + \tau_2 x^2 + \tau_1 x + \tau_0)^3 = \tau_5^3 x^{15} + \tau_4^3 x^{12} + \tau_3^3 x^9 + \tau_2^3 x^6 + \tau_1^3 x^3 + \tau_0^3$, cubing modulo $f_{g^a}(x)$ costs at least $6C_1 + 21M_1$. Moreover, using Karatsuba's technique, multiplying two elements in $\mathbb{F}_{q^6}$ appears to require at least $18M_1$ in $\mathbb{F}_q$. Hence computing $c_{ab} = \mathrm{Tr}(x^b)$ costs *at least* $2A_1 + 1F_1 + 1S_1 + (6C_1 + 21M_1 + \frac{36}{(2w+1)}M_1)\ell$.

6. Comparisons

6.1. **Factor-4 compression.** We compare the running times of the five exponentiation algorithms presented in Section 4. We first analyze the costs $SR_{2,1}$ and $SR_{2,2}$. Recall that $SR_{2,1}$ is the cost of finding a root of the irreducible polynomial $\tilde{f}(x) = x^2 + cx + c^t$ for some $c \in \mathbb{F}_q$, and $SR_{2,2}$ is the cost of finding a root of the irreducible polynomial $\tilde{f}(x) = x^2 + cx + 1$ for some $c \in \mathbb{F}_{q^2}$. Shoup [26] showed that if a square-free degree-m polynomial over $\mathbb{F}_{p^l}$, where p is a small prime, is known to factor into d distinct irreducible same-degree polynomials over $\mathbb{F}_{p^l}$, then the factorization of that polynomial can be obtained at a cost of $\tilde{O}(m(dl)^2)$ $\mathbb{F}_{p^l}$-operations. Therefore, we may set $SR_{2,1} = \tilde{O}(8(\log_2 q)^2)$ $\mathbb{F}_{q^2}$-operations, and $SR_{2,2} = \tilde{O}(32(\log_2 q)^2)$ $\mathbb{F}_{q^4}$-operations.

Table 6.1. Comparison of exponentiation algorithms for factor-4 compression. The exponent is an ℓ-bit integer.

Algorithms	Precomputation	Main Loop
Algorithm 1	$1I_1 + 1M_1$	$(4M_1 + 4S_1)(\ell - 1)$
Algorithm 2	$1SR_{2,1}$	$(1M_2 + 1S_2)(\ell - 1)$
DDE (Section 4.3.2)	$1SR_{2,1} + 1SR_{2,2}$	$(\frac{9}{(w+1)}M_1 + 4S_1)\ell$
BPV-I (Section 4.3.3)	$1SR_{2,1}$	$(\frac{3}{2}M_2 + 2S_2)(\ell - 1)$
BPV-II (Section 4.3.4)	$1F_1$	$(7M_1 + 4S_1)(\ell - 1)$

We can conclude from Table 6.1 that, for a suitable choice of w, Algorithm DDE is the fastest algorithm if the precomputations are done in advance (which is the case if the base is fixed). Otherwise, Algorithm 1 is the fastest one.

Remark 6.1. It is possible to obtain better running time estimates (at least for the running times of the main loops) for some of the algorithms listed in Table 6.1. Adapting the double-exponentiation technique given in [28] to speed up Algorithm 2, one can estimate the running time of Algorithm 2 as $2SR_{2,1} + (0.75M_2)\ell$. Using simultaneous multi-exponentiation or window-NAF techniques one can optimize the running times of Algorithms BPV-I and BPV-II. These techniques do not seem to reduce the cost of the squaring steps in the main loops of Algorithm BPV-I and BPV-II (which are $(1M_2 + 2S_2)(\ell - 1)$ and $(6M_1 + 4S_1)(\ell - 1)$, respectively) other than transferring some of the cost to the precomputation phase. The mixed-multiplication method (see [13]) can be adapted to speed up Algorithm DDE given input values c_a and b. More precisely, after decompressing c_a to g^a, one can first compute $g^{ab}h$ for some $h \in \mathbb{F}_{q^2}^*$ at an approximate cost of $(\frac{2}{(w+1)}M_2)\ell$ instead of computing g^{ab} using Karatsuba's multiplication technique at an approximate cost of $(\frac{9}{(w+1)}M_1)\ell$ (see Section 4.3.2), Then c_{ab} can be computed by taking the square root of $\text{Tr}((g^{ab}h)^{q^2-1}) = \text{Tr}(g^{ab(q^2-1)}) = c_{2ab}$. Using this method, the cost of Algorithm DDE can be approximated as $1SR_{2,1} + 1SR_{2,2} + (\frac{6}{(w+1)}M_1)\ell + 1I_4$; the cost of this faster algorithm is given in Table 6.2. Hence, we still expect Algorithm 1 to be the fastest algorithm for general bases and, if the base is fixed, Algorithm DDE to be faster than Algorithm 1 and Algorithm 2 for a suitable choice of w.

To be more concrete, we list the expected running times of the five exponentiation algorithms in a particular setting in Table 6.2 based on the estimates given in Table 6.1. For the 128-bit security level, we let $q = 2^{1223}$ and $t = 2^{612}$. Then $q + 1 + t = 5n$ where n is a 1221-bit prime. We will ignore the costs F_i, S_i and, using Karatsuba's technique, assume $M_2 = 3M_1$. We also select $w = 5$.

TABLE 6.2. Comparison of exponentiation algorithms for factor-4 compression at the 128-bit security level. The exponent is an 1221-bit integer.

Algorithms	Multiplication cost for general bases	Multiplication cost for fixed bases
Algorithm 1	$1I_1 + 4881M_1$	$4880M_1$
Algorithm 2	$1SR_{2,1} + 3660M_1$	$3660M_1$
DDE (Section 4.3.2, Remark 6.1)	$1SR_{2,1} + 1SR_{2,2} + 1I_4 + 1221M_1$	$1I_4 + 1221M_1$
BPV-I (Section 4.3.3)	$1SR_{2,1} + 5490M_1$	$5490M_1$
BPV-II (Section 4.3.4)	$8540M_1$	$8540M_1$

6.2. **Factor-6 compression.** We compare the running times of the six algorithms presented in Section 5 and the XTR$_3$ [25] algorithm. We note that even though the running time of XTR$_3$ is estimated as $1SR_{2,1} + (8M_1)\ell$ in [25], adapting the double-exponentiation technique for speeding up XTR (see [28]) to XTR$_3$ one can roughly approximate the cost of XTR$_3$ as $2SR_{2,1} + (3M_1)\ell$ which we use it in Table 6.3.

We first analyze the costs $SR_{2,1}$, $SR_{2,3}$ and $SR_{3,1}$. Similarly as in Section 6.1, using Shoup's method [26] we may let $SR_{3,1} = \tilde{O}(27(\log_2 q)^2)$ $\mathbb{F}_{q^3}$-operations. Now, we will consider $SR_{2,1}$ and $SR_{2,3}$, and see that these costs are negligible comparing to $SR_{3,1}$. We let $q = 3^{2r+1}$ and recall that $SR_{2,1}$ is the cost of finding a root of the irreducible polynomial $\tilde{f}(x) = x^2 - cx + c^t$ over $\mathbb{F}_q$. The roots of this polynomial are $2(c \pm \sqrt{c^2 + 2c^t})$. Since $\tilde{f}(x)$ is irreducible over $\mathbb{F}_q$, $C = c^2 + 2c^t$ must be a quadratic non-residue in $\mathbb{F}_q$. Moreover, since $q \equiv 3 \pmod 4$, -1 is a quadratic non-residue in $\mathbb{F}_q$. Therefore, $-C$ is a quadratic residue in $\mathbb{F}_q$ and finding a root of $\tilde{f}(x)$ reduces to finding a square root of $-C$ in $\mathbb{F}_q$. Namely, the roots of $\tilde{f}(x)$ are $2(c \pm \sqrt{-C}\sqrt{-1})$ and also note that $\sqrt{-C} = (-C)^{\frac{q+1}{4}}$ and $\sqrt{-1} \in \mathbb{F}_{q^2}$. Barreto et al. [2] observed that

$$\frac{q+1}{4} = 6\sum_{i=0}^{r-1}(3^2)^i + 1$$

and that computing

$$(-C)^{\frac{q+1}{4}} = \left((C^2)^{\sum_{i=0}^{r-1}(3^2)^i}\right)^3(-C)$$

can be achieved at a cost of $1S_1 + (\lfloor\log_2 r\rfloor + HW(r))M_1$. Therefore, after computing $\sqrt{-1} \in \mathbb{F}_{q^2}$ we can compute a root of $\tilde{f}(x)$ in $\mathbb{F}_{q^2}$ at a cost of $1m_2 + 1M_2 + 1S_1 + (\lfloor\log_2 r\rfloor + HW(r) + 1)M_1$. Hence, we may set $1SR_{2,1} = 1m_2 + 1M_2 + 1S_1 + (\lfloor\log_2 r\rfloor + HW(r) + 1)M_1$. Similarly, we can show $1SR_{2,3} = 1m_6 + 1M_6 + 1S_3 + (\lfloor\log_2(3r+1)\rfloor + HW(3r+1) + 1)M_3$.

Table 6.3. Comparison of exponentiation algorithms for factor-6 compression. The exponent is an ℓ-bit integer.

Algorithms	Precomputation	Main Loop
Algorithm 3	$1I_1 + 2M_1 + 2S_1$	$(53A_1 + 6F_1 + 23M_1 + 6S_1)(\ell - 1)$
Algorithm 4	$1SR_{3,1}$	$(1A_3 + 2C_3 + 2M_3)(\log_3 2)(\ell - 1)$
DDE (Section 5.3.2)	$1SR_{3,1} + 1SR_{2,3}$	$(6C_1 + \frac{36}{(2w+1)}M_1)(\log_3 2)\ell$
BPV-I (Section 5.3.3)	$1SR_{2,1}$	$(\frac{17}{3}A_2 + 3C_2 + 7M_2)(\log_3 2)(\ell - 1)$
BPV-II (Section 5.3.4)	$1SR_{3,1}$	$(1A_1 + 2C_3 + 2M_3 + \frac{8}{(2w+1)}M_3)(\log_3 2)\ell$
BPV-III (Section 5.3.5)	$1S_1$	$\geq (6C_1 + 21M_1 + \frac{36}{2w+1}M_1)(\log_3 2)\ell$
XTR$_3$ ([25])	$2SR_{2,1}$	$(3M_1)\ell$

We can conclude from Table 6.3 that, for a suitable choice of w, Algorithm DDE is the fastest algorithm if the precomputations are done in advance (which is the case if the base is fixed). Otherwise, XTR$_3$ is the fastest one.

Remark 6.2. It is possible to obtain better running time estimates (at least for the running times of the main loops) for some of the algorithms listed in Table 6.3. Adapting the double-exponentiation technique given in [28] to speed up Algorithm 4, one can estimate the running time of Algorithm 4 as $2SR_{3,1} + (0.75M_3 + 0.17S_3)\ell$. Using simultaneous multi-exponentiation or window-NAF techniques one might optimize the running times of Algorithms BPV-I, BPV-II and BPV-III. These techniques do not seem to reduce the cost of the cubing steps in the main loops of Algorithms BPV-I, BPV-II and BPV-III (which are roughly $(5M_2)(\log_3 2)\ell \approx (6.31M_1)\ell$, $(2M_3)(\log_3 2)\ell \approx (7.57M_1)\ell$ and $(21M_1)(\log_3 2)\ell \approx (13.25M_1)\ell$, respectively) other than transferring some of the cost to the precomputation phase. The mixed-multiplication method (see [13]) can be adapted to speed up Algorithm DDE by slightly changing its output value from c_{ab} to c_{2ab} given input values c_a and b (we note that Algorithm DDE with this slight change in its output can still be used in the cryptographic applications mentioned in Section 2). More precisely, after decompressing c_a to g^a, one can first compute $g^{ab}h$ for some $h \in \mathbb{F}_{q^3}^*$ at an approximate cost of $(\frac{4}{(2w+1)}M_3)(\log_3 2)\ell$ instead of computing g^{ab} using Karatsuba's multiplication technique at an approximate cost of $(\frac{36}{(2w+1)}M_1)(\log_3 2)\ell$ (see Section 5.3.2). Then $\mathrm{Tr}((g^{ab}h)^{q^3-1}) = \mathrm{Tr}(g^{ab(q^3-1)}) = c_{2ab}$ can be computed at an approximate cost of $1I_6$. Using this method, the cost of Algorithm DDE can be approximated as $1SR_{3,1} + 1SR_{2,3} + (\frac{24}{(2w+1)}M_1)(\log_3 2)\ell + 1I_6$; the cost of this faster algorithm is given in Table 6.4. Hence, we still expect XTR$_3$ to be the fastest algorithm for general bases and, if the base is fixed, Algorithm DDE to be significantly faster than the other algorithms for a suitable choice of w.

To be more concrete, we list the expected running times of the seven exponentiation algorithms in a particular setting in Table 6.4 based on the estimates given in Table 6.3. For the 128-bit security level, we let $q = 3^{509}$ and $t = 3^{255}$. Then $q+1-t = 7n$ where n is an 804-bit prime. We will ignore the costs A_i, a_i, C_i, F_i, m_i and, using Karatsuba's technique, assume $M_2 = 3M_1$, $M_3 = 6M_1$ and $M_6 = 18M_1$. We also assume $S_i = M_i$, and select $w = 5$. Note that $r = 254$, $\lfloor \log_2 r \rfloor = 8$, $\lfloor \log_2 (3r+1) \rfloor = 9$, $HW(r) = 7$, $HW(3r+1) = 8$, and $1SR_{2,1} = 20M_1$ and $1SR_{2,3} = 132M_1$.

Table 6.4. Comparison of exponentiation algorithms for factor-6 compression at the 128-bit security level. The exponent is an 804-bit integer.

Algorithms	Multiplication cost for general bases	Multiplication cost for fixed bases
Algorithm 3	$1I_1 + 23291M_1$	$23287M_1$
Algorithm 4	$1SR_{3,1} + 6080M_1$	$6080M_1$
DDE (Section 5.3.2, Remark 6.2)	$1SR_{3,1} + 1I_6 + 1239M_1$	$1I_6 + 1107M_1$
BPV-I (Section 5.3.3)	$10660M_1$	$10640M_1$
BPV-II (Section 5.3.4)	$1SR_{3,1} + 8301M_1$	$8301M_1$
BPV-III (Section 5.3.5)	$\geq 12314M_!$	$\geq 12313M_1$
XTR$_3$ ([25])	$2452M_1$	$2412M_1$

7. Concluding remarks

We have shown how to compress, by a factor of 4, pairing values of the commonly-used symmetric bilinear pairings over characteristic two fields, and also further explored compressing pairing values of symmetric bilinear pairings over characteristic three fields by a factor of 6. We have shown how one can exponentiate using the compressed pairing values. Our exponentiation algorithms are reasonably efficient. In particular, if the base is fixed then we expect at least a 54% speed up over the fastest previously known algorithm XTR$_3$ for the factor-6 compression case. Finding more efficient algorithms would be worthwhile. It would also be desirable to implement the algorithms to verify our estimates of their relative efficiency.

Acknowledgments

I thank Alfred Menezes for generously spending his time with me throughout my work, and for his suggestions that significantly enriched the paper. I also thank the two anonymous referees for their detailed and very helpful comments.

References

1. O. Ahmadi, D. Hankerson, and A. Menezes, *Software implementation of arithmetic in* $\mathbb{F}_{3^m}$, International Workshop on Arithmetic of Finite Fields – WAIFI 2007, Lecture Notes in Computer Science **4547** (2007), 85–102.
2. P. Barreto, H. Kim, B. Lynn, and M. Scott, *Efficient algorithms for pairing based cryptosystems*, Advances in Cryptology – CRYPTO 2002, Lecture Notes in Computer Science **2442** (2002), 354–369.
3. P. Barreto and M. Naehrig, *Pairing-friendly elliptic curves of prime order*, Selected Areas in Cryptography – SAC 2005, Lecture Notes in Computer Science **3897** (2006), 319–331.
4. D. Bleichenbacher, W. Bosma, and A. Lenstra, *Some remarks on lucas-based cryptosystems*, Advances in Cryptology – CRYPTO'95, Lecture Notes in Computer Science **936** (1995), 306–316.
5. D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, SIAM Journal on Computing **32** (2003), 586–615.
6. A. Brouwer, R. Pellikaan, and E. Verheul, *Doing more with fewer bits*, Advances in Cryptology – ASIACRYPT '99, Lecture Notes in Computer Science **1716** (1999), 321–332.
7. D. Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Transactions on Information Theory **30** (1984), 587–594.

8. W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **22** (1976), 644–65.
9. FIPS 186-3, *Digital signature standard (DSS)*, Federal Information Processing Standards Publication 186-3, National Institute of Standards and Technology, 2009.
10. K. Giuliani and G. Gong, *Analogues to the Gong-Harn and XTR cryptosystems*, Technical Report CORR 2003-34, available at `http://www.cacr.math.uwaterloo.ca/techreports/2003/corr2003-34.ps`.
11. G. Gong and L. Harn, *Public-key cryptosystems based on cubic finite field extensions*, IEEE Transactions on Information Theory **45** (1999), no. 7, 2601–2605.
12. D. Gordon, *Discrete logarithms in $GF(p)$ using the number field sieve*, SIAM Journal on Discrete Mathematics **6** (1993), 124–138.
13. R. Granger, D. Page, and M. Stam, *On small characteristic algebraic tori in pairing-based cryptography*, LMS Journal of Computation and Mathematics **9** (2004), 64–85.
14. A. Joux, *A one round protocol for tripartite Diffie-Hellman*, ANTS-IV: Algorithmic Number Theory 4th International Symposium, Lecture Notes in Computer Science **1838** (2000), 385–393.
15. N. Koblitz, *An elliptic curve implementation of the finite field digital signature algorithm*, Advances in Cryptology – CRYPTO '98, Lecture Notes in Computer Science **1462** (1998), 327–337.
16. A. Lenstra, *Unbelievable security. matching AES security using public key systems*, Advances in Cryptology – ASIACRYPT 2001, Lecture Notes in Computer Science **2248** (2001), 67–86.
17. A. Lenstra and E. Verheul, *The XTR public key system*, Advances in Cryptology – CRYPTO 2000, Lecture Notes in Computer Science **1880** (2000), 1–19.
18. P. Montgomery, *Evaluating recurrences of form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas chains*, `www.cwi.nl/ftp/pmontgom/Lucas.ps.gz`, December 13, 1983; Revised March, 1991 and January, 1992.
19. J. Muir and D. Stinson, *Minimality and other properties of the width-w nonadjacent form*, Mathematics of Computation **75** (2005), 369–384.
20. J. Pollard, *Monte Carlo methods for index computation mod p*, Mathematics of Computation **32** (1978), 918–924.
21. K. Rubin and A. Silverberg, *Torus-based cryptography*, Advances in Cryptology – CRYPTO 2003, Lecture Notes in Computer Science **2729** (2003), 349–365.
22. R. Sakai, T. Ohgishi, and M. Kasahara, *Cryptosystems based on pairings*, Symposium on Cryptography and Information Security – SCIS 2000, Okinawa, Japan (2000), 26–28.
23. 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/`.
24. M. Scott and P. Barreto, *Compressed pairings*, Advances in Cryptology – CRYPTO 2004, Lecture Notes In Computer Science **3152** (2004), 140–156.
25. 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** (2008), no. 10, 2843–2850.
26. V. Shoup, *A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic*, ISSAC '91: International Symposium on Symbolic and Algebraic Computation, Association for Computing Machinery (1991), 14–21.
27. 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** (1994), 357–364.
28. M. Stam and A. Lenstra, *Speeding up XTR*, Advances in Cryptology – ASIACRYPT 2001, Lecture Notes in Computer Science **2248** (2001), 125–143.
29. T. Takagi, S. Yen, and B. Wu, *Radix-r non-adjacent form*, Information Security – ISC 2004, Lecture Notes In Computer Science **3225** (2004), 99–110.
30. 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** (2005), 234–250.
31. M. van Dijk and D. Woodruff, *Asymptotically optimal communication for torus-based cryptography*, Advances in Cryptology – CRYPTO'2004, Lecture Notes in Computer Science **3152** (2004), 151–178.

32. B. Waters, *Efficient identity-based encryption without random oracles*, Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science **3494** (2005), 114–127.

Appendix A. Previous work on compression factors 2,3 and 6

A.1. **Compression factor 2 (LUC cryptosystem).** Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{q^2}^*$ of order $n = q+1$. Let $c_u = \operatorname{Tr}(g^u)$ be the trace of g^u over $\mathbb{F}_q$, and $f_{g^u}(x)$ the minimal polynomial of g^u over $\mathbb{F}_q$. Then $f_{g^u}(x) = x^2 - c_u x + 1$ and the following recursive relation holds for all integers u and v:

$$c_{u+v} = c_u c_v - c_{u-v}.$$

In particular, $c_{2u} = c_u^2 - c_0 = c_u^2 - 2$ and $c_{2u+1} = c_{u+1}c_u - c_1$, whence given a state $s_u = [c_u, c_{u+1}]$ one can compute s_{2u} and s_{2u+1}. This observation leads to an efficient double-and-add algorithm for computing c_{ab} given c_a and b (see [27] and [4]). The cost of the algorithm is $\log_2 b$ steps, each step costing 1 multiplication and 1 squaring in $\mathbb{F}_q$.

A.2. **Compression factor 3 (LUCKY cryptosystem).** Let $\mu_n = \langle g \rangle$ be the multiplicative subgroup of $\mathbb{F}_{p^6}^*$ of order $n = p^2 - p + 1$. Brouwer, Pellikaan and Verheul [6] obtained a compression factor 3 by identifying elements of μ_n with the coefficients of their minimal polynomials over $\mathbb{F}_p$. In particular, each element g^a can be uniquely identified (up to conjugation over $\mathbb{F}_p$) with a pair $(c_a, d_a) \in \mathbb{F}_p \times \mathbb{F}_p$. They also presented an algorithm (see Sections 3.3 and 5 in [6]) to compute (c_{ab}, d_{ab}) given (c_a, d_a) and an integer b. The algorithm is as follows. Given (c_a, d_a) we first construct the minimal polynomial of g^a over $\mathbb{F}_p$ and adjoin a root ρ_a of this polynomial to $\mathbb{F}_p$ thus obtaining a copy of $\mathbb{F}_{p^6}$. Next, we can raise ρ_a to the power b and determine the minimal polynomial of ρ_a^b over $\mathbb{F}_p$. From the coefficients of this polynomial we can determine (c_{ab}, d_{ab}), as required.

A.3. **Compression factor 3 (XTR cryptosystem).** Let $p \equiv 2 \pmod 3$ be a prime and $\mu_n = \langle g \rangle$ the multiplicative subgroup of $\mathbb{F}_{p^6}^*$ of order $n = p^2 - p + 1$. Let $c_u = \operatorname{Tr}(g^u)$ be the trace of g^u over $\mathbb{F}_{p^2}$, and $f_{g^u}(x)$ the minimal polynomial of g^u over $\mathbb{F}_{p^2}$. It was observed in [17] that $f_{g^u}(x) = x^3 - c_u x^2 + c_u^p x - 1$ and the following recursive relation holds for all integers u and v:

$$c_{u+v} = c_u c_v - c_v^p c_{u-v} + c_{u-2v}.$$

In particular, it was shown in [17] that given a state $s_u = [c_{u-1}, c_u, c_{u+1}]$ one can compute s_{2u} and s_{2u+1}. This observation leads to an efficient double-and-add algorithm for computing c_{ab} given c_a and b.

A.4. **Compression factor 6 (XTR$_3$ cryptosystem).** Let $q = 3^{2r+1}$ and $\mu_n = \langle g \rangle$ the multiplicative subgroup of $\mathbb{F}_{q^6}^*$ of order $n = q - \sqrt{3q} + 1$. Let $c_u = \operatorname{Tr}(g^u)$ be the trace of g^u over $\mathbb{F}_q$. It was observed in [25] that given c_u one can efficiently compute the trace $\tilde{c}_u$ of g^u over $\mathbb{F}_{q^2}$ up to conjugation over $\mathbb{F}_q$. Now, given c_a and b, one can first compute $\tilde{c}_a$, then compute $\tilde{c}_{ab}$ using an algorithm analogous to that of XTR, and finally obtain $c_{ab} = \tilde{c}_{ab} + \tilde{c}_{ab}^q$. The overall cost is $8\log_2 b + \lfloor \log_2 r \rfloor + HW(r) + 2$ multiplications in $\mathbb{F}_q$, where $HW(\cdot)$ denotes the Hamming weight function.

Appendix B. Non-uniqueness of factor-6 compression

We list some parameters (p, n, T, g, u, v) where $p > 3$ and n are prime, $n \mid (p^4 - p^2 + 1)$, and if $\mu_n = \langle g \rangle$ is the subgroup of $\mathbb{F}_{q^6}^*$ (where $q = p^2$) of embedding degree 6 over $\mathbb{F}_q$, then the trace function $\mathrm{Tr} : \mu_n \to \mathbb{F}_q$ has collisions with collision value $T = \mathrm{Tr}(g^u) = \mathrm{Tr}(g^v)$, where g^u and g^v are not conjugates over $\mathbb{F}_q$. In fact, the first example corresponds to the parameter of a BN curve (see Remark 5.2 and [3]). The subgroups μ_n in examples (2)–(5) can be realized as the images of pairing functions defined on elliptic curves over $\mathbb{F}_p$ having embedding degree 12; these elliptic curves are not BN curves. On the other hand, the subgroups μ_n in examples (6)–(9) cannot be realized as the image of pairing functions defined on elliptic curves over $\mathbb{F}_p$ since n is not in the Hasse interval $[(\sqrt{p} - 1)^2, (\sqrt{p} + 1)^2]$.

(1) $(55333, 55117, 45541, g, 2583, 6758)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 2)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 51894z + 9346)$,
$g = (5638z + 51877)w^5 + (13297z + 52777)w^4 + (20924z + 25318)w^3 + (12991z + 51370)w^2 + (12014z + 15762)w + 15570z + 33355$.

(2) $(113, 97, 46, g, 20, 29)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 101z + 3)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 112z)$,
$g = (77z + 47)w^5 + (29z + 36)w^4 + (52z + 24)w^3 + (58z + 14)w^2 + (70z + 19)w + 35z + 49$.

(3) $(297457, 296941, 170970, g, 42243, 120695)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 5)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 226781z + 185746)$,
$g = (16282z + 114368)w^5 + (264807z + 131493)w^4 + (52866z + 175278)w^3 + (153254z + 81017)w^2 + (55521z + 87692)w + 27500z + 23791$.

(4) $(757363, 758053, 147442, g, 195883, 532217)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 1)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 538552z + 38070)$,
$g = (155890z + 54538)w^5 + (593065z + 407753)w^4 + (421831z + 252766)w^3 + (260748z + 405992)w^2 + (426293z + 142508)w + 240615z + 248519$.

(5) $(758743, 758053, 181973, g, 26808, 304248)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 1)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 205464z + 531602)$,
$g = (644749z + 587471)w^5 + (152111z + 593113)w^4 + (218499z + 561168)w^3 + (605298z + 638869)w^2 + (634695z + 684366)w + 667616z + 318403$.

(6) $(107873, 100333, 10836z + 78750, g, 6775, 11682)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 3)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 70681z + 104404)$,
$g = (74900z + 35768)w^5 + (67288z + 57726)w^4 + (107242z + 94650)w^3 + (31629z + 3630)w^2 + (87341z + 35135)w + 64176z + 45731$.

(7) $(107873, 100333, 97037z + 78750, g, 14995, 20801)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 3)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 70681z + 104404)$,
$g = (71680z + 68567)w^5 + (99591z + 66980)w^4 + (34944z + 30340)w^3 + (8164z + 61554)w^2 + (29313z + 44640)w + 33137z + 62160$.

(8) $(147347, 135193, 56095z + 80249, g, 4989, 12193)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 1)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 76671z + 35636)$,
$g = (120469z + 82203)w^5 + (77634z + 4734)w^4 + (127289z + 74128)w^3 + (106306z + 13444)w^2 + (82983z + 115891)w + 34710z + 136734$.

(9) $(147347, 135193, 91252z + 80249, g, 3676, 12104)$ where
$\mathbb{F}_q = \mathbb{F}_p[z]/(z^2 + 1)$,
$\mathbb{F}_{q^6} = \mathbb{F}_q[w]/(w^6 + 76671z + 35636)$,
$g = (113834z + 48691)w^5 + (87284z + 70855)w^4 + (85568z + 73528)w^3 + (102712z + 53673)w^2 + (13537z + 46246)w + 105305z + 14472$.

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

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