SQUARING IN CYCLOTOMIC SUBGROUPS

KORAY KARABINA

Abstract. We propose new squaring formulae for cyclotomic subgroups of certain finite fields. Our formulae use a compressed representation of elements having the property that decompression can be performed at a very low cost. The squaring formulae lead to new exponentiation algorithms in cyclotomic subgroups which outperform the fastest previously-known exponentiation algorithms when the exponent has low Hamming weight. Our algorithms can be adapted to accelerate the final exponentiation step of pairing computations.

1. Introduction

One challenge in cryptography is to achieve a desired level of security in the most efficient way. The efficiency can generally be improved if one can implement a cryptosystem with more compact parameters. An example in the context of discrete logarithm cryptosystems is the performance of 256-bit elliptic curve groups defined over 256-bit finite fields versus 256-bit subgroups of 3072-bit finite field groups. Even though both systems are believed to provide 128-bit security with a careful choice of parameters, the former leads to a more efficient implementation than the latter mainly because the points in the corresponding elliptic curve group are represented with fewer bits than the elements in the subgroup of the corresponding finite field group.

In recent years, there have been several proposals to represent the elements of cyclotomic subgroups of finite field groups with fewer bits than is required in their natural representation and to compute with the compressed representation of elements [26, 9, 5, 18, 8, 22, 30, 29, 25, 15, 16, 27, 14]. These methods help close the gap between the efficiency of elliptic curve cryptosystems and finite field based cryptosystems. A related research objective is to improve the efficiency of finite field arithmetic using the special structure of cyclotomic subgroups. The most recent work is by Granger and Scott [12] who improved and extended the results in [10] and [28]. They showed that if $q \equiv 1 \pmod 6$ then the squaring operation in the order-(q^2-q+1) cyclotomic subgroup of $\mathbb{F}_{q^6}^*$ can be performed at a cost of only 6 multiplications in $\mathbb{F}_q$ (or 3 squarings in $\mathbb{F}_{q^2}$).

We should emphasize that there are squaring algorithms that work with compressed representations of elements and that are faster than the method proposed in [12] for the order-(q^2-q+1) cyclotomic subgroup G of $\mathbb{F}_{q^6}^*$. For example, if $g \in G$ then one can adapt the XTR technique [18] to compute $\mathrm{Tr}_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}(g^2)$ from $\mathrm{Tr}_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}(g)$ at a cost dominated by 2 multiplications in $\mathbb{F}_q$. However, since the trace function is not multiplicative, many cyptographic protocols that require multiplying more than two elements in G do not seem to benefit from this fast trace-based squaring method. In particular, given only the traces of elements, the cost of recovering the full representation of these elements (decompression) and also the cost of multiplication with the trace representation greatly dominate the cost of multiplying two elements given in their natural representations in

Key words and phrases. Cyclotomic subgroups, squaring, pairing-based cryptography.

$\mathbb{F}_{q^6}$. In addition, even though the trace-based methods yield single-exponentiation and double-exponentiation algorithms that are faster than their conventional counterparts, it is not known how to use trace-based methods in general multi-exponentiation algorithms. Therefore, it is natural to look for squaring formulae that work with compressed representations and can be effectively incorporated into cryptographic applications.

In Section 2 we classify the known squaring techniques for cyclotomic subgroups in two categories: those that work with natural representation of elements, and those that work with compressed representation of elements. We will focus on the order $(q^2 - q + 1)$-cyclotomic subgroup G of $\mathbb{F}_{q^6}^*$ as this seems to be the most interesting case with respect to some potential applications in pairing-based cryptography. In Section 3, we present a new formula for squaring elements in G when $q \equiv 1 \pmod 6$. We first describe a function $\mathcal{C}$ that compresses elements $g \in G$ by a factor of 3/2, and describe a decompression function $\mathcal{D}$ that can be computed very efficiently and satisfies $\mathcal{D}(\mathcal{C}(g)) = g$ for all $g \in G$. Our new squaring method works with this compressed form of elements. Given $\mathcal{C}(g)$, the cost of computing $\mathcal{C}(g^2)$ is dominated by 4 multiplications in $\mathbb{F}_q$ (or 2 squarings in $\mathbb{F}_{q^2}$). Note that this is 33% faster than the method described in [12]. The efficient decompression will permit us to effectively utilize the squaring formula in exponentiation algorithms, especially when the exponent has low Hamming weight. In Section 4, we discuss some applications of our squaring formula and provide some comparisons based on operation counts. In Section 5, we describe a more general technique to search for efficient squaring formulae. As a result, we discover other squaring formulae some of which seem to offer better performance in particular cases. Section 5 also shows that some of the previously-known squaring formulae can be obtained via our search method. In Section 6, we compare our new squaring formulae and the squaring formulae in [12]. We conclude in Section 7.

2. A review of squaring and exponentiation algorithms in cyclotomic subgroups of $\mathbb{F}_{q^6}^*$

Let $\mathbb{F}_q$ be a finite field with q elements and characteristic not equal to 2 or 3. For simplicity, we first assume that $q = p$ is prime. We denote by $G_{\Phi_6(p)}$ the order-$\Phi_6(p)$ cyclotomic subgroup of $\mathbb{F}_{p^6}^*$. Here, $\Phi_i(p)$ denotes the ith cyclotomic polynomial evaluated at p, and $|G_s| = s$. Note that $|G_{\Phi_6(p)}| = p^2 - p + 1$. Since $\mathbb{F}_{p^6}$ is the smallest extension of $\mathbb{F}_p$ that contains $G_{\Phi_6(p)}$, an element $g \in G_{\Phi_6(p)}$ is naturally represented with 6 $\mathbb{F}_p$-elements. However, exploiting the algebraic structure of $G_{\Phi_6(p)}$, one can represent $g \in G_{\Phi_6(p)}$ with 3 or even with 2 $\mathbb{F}_p$-elements yielding more compact representations by factor 2 or 3. Trace-based compression and torus-based compression are the two known methods to achieve factor 2 and 3 compression in $G_{\Phi_6(p)}$ [26, 18, 22]. We summarize next the fastest previously-known squaring algorithms in $G_{\Phi_6(p)}$. These algorithms fall into two categories: those that work with compressed representation of elements, and those that work with full representation of elements.

2.1. **Compressed representations.**

2.1.1. ***Trace-based squaring:*** Let Tr_{p^i,p^j} denote the trace function $\mathrm{Tr}_{\mathbb{F}_{p^i}/\mathbb{F}_{p^j}} : \mathbb{F}_{p^i} \to \mathbb{F}_{p^j}$. Elements $g \in G_{\Phi_6(p)}$ can be uniquely represented by their traces $\mathrm{Tr}_{p^6,p^3}(g)$ (upto conjugation over $\mathbb{F}_{p^3}$) [26], or $\mathrm{Tr}_{p^6,p^2}(g)$ (upto conjugation over $\mathbb{F}_{p^2}$) [18]. More interestingly, one can compute $\mathrm{Tr}_{p^6,p^3}(g^2)$ and $\mathrm{Tr}_{p^6,p^2}(g^2)$ given $\mathrm{Tr}_{p^6,p^3}(g)$ and $\mathrm{Tr}_{p^6,p^2}(g)$, respectively.

The corresponding squaring algorithms are known as LUC squaring and XTR squaring, respectively.

LUC-squaring: $\mathrm{Tr}_{p^6,p^3}(g^2) = \mathrm{Tr}_{p^6,p^3}(g)^2 - 2$. The cost of LUC-squaring is dominated by 1 squaring in $\mathbb{F}_{p^3}$.

XTR-squaring: $\mathrm{Tr}_{p^6,p^2}(g^2) = \mathrm{Tr}_{p^6,p^2}(g)^2 - 2\mathrm{Tr}_{p^6,p^2}(g)^p$. The cost of XTR-squaring is dominated by 1 squaring in $\mathbb{F}_{p^2}$ (since the cost of the Frobenius $c \mapsto c^p$ is negligible for $c \in \mathbb{F}_{p^2}$).

2.1.2. ***Torus-based squaring:*** Let $\mathbb{F}_{p^6} = \mathbb{F}_{p^3}(\sigma)$ where σ is a root of $x^2 - c$ for some quadratic non-residue $c \in \mathbb{F}_{p^3}$. Elements $g = g_0 + g_1\sigma \in G_{\Phi_6(p)} \setminus \{\pm 1\}$ can be uniquely represented by $\alpha = (g_0 + 1)/g_1 \in \mathbb{F}_{p^3}$. In fact, if $\alpha \in \mathbb{F}_{p^3}$ is the compact representation of $g \in G_{\Phi_6(p)}$ then $g = (\alpha + \sigma)/(\alpha - \sigma)$ and $g^2 = (\alpha^2 + c + 2\alpha\sigma)/(\alpha^2 + c - 2\alpha c)$; see [23]. Now, given $g^{2^i} = (x + y\sigma)/(x - y\sigma) \in G_{\Phi_6(p)}$ for some $x, y \in \mathbb{F}_{p^3}$, one can write $g^{2^{i+1}} = (x^2 + y^2c + 2xy\sigma)/(x^2 + y^2c - 2xy\sigma)$. In order to avoid the inversion operation in the squaring formula, one can encode an element $g = (x + y\sigma)/(x - y\sigma) \in G_{\Phi_6(p)} \setminus \{\pm 1\}$ with $\mathcal{P}(g) = [x, y]$ and compute $\mathcal{P}(g^2) = [x^2 + y^2c, 2xy]$, the (unique) representative of g^2. This is so-called squaring with *mixed/projective* coordinates; see, for example, [11]. We call this the TB2-squaring method as it uses factor-2 torus-based compression.

TB2-squaring: $\mathcal{P}(g^2) = [x^2 + y^2c, 2xy]$, where $\mathcal{P}(g) = [x, y]$. The cost of TB2-squaring is dominated by 2 multiplications in $\mathbb{F}_{p^3}$ because $x^2 + y^2c = (x + yc)(x + y) - (c + 1)xy$ and we can ignore the cost of addition, subtraction, and multiplication by c.

Remark 2.1. Given $g = (\alpha + \sigma)/(\alpha - \sigma) \in G_{\Phi_6(p)}$ for some $\alpha \in \mathbb{F}_{p^3}$, one can exploit the algebraic structure of $G_{\Phi_6(p)}$ to further compress α to two $\mathbb{F}_p$-elements [22, 11]. One can then define TB3-squaring analogous to TB2-squaring. However, to the author's knowledge, this extra structure hast not yet been exploited to derive efficient TB3-squaring formulae. For example, it was reported in [11] that the cost of a TB3-squaring that uses the factor-3 torus-based compressed representation of elements in $G_{\Phi_6(p)}$ is 21 multiplications, 38 additions and 1 inversion in $\mathbb{F}_p$ (when $p \equiv 2, 5 \pmod 9$). Even though the inversion can be eliminated by using mixed/projective coordinates as in TB2-squaring, the cost is still dominated by 21 multiplications in $\mathbb{F}_p$, which is more expensive than the above TB2-squaring.

2.2. **Full representations.** The squaring formulae that work with the natural representation of g can be summarized as below.

General squaring: Let $\mathbb{F}_{p^6} = \mathbb{F}_{p^3}(\sigma)$, where σ is a root of $\sigma^2 - c$ for some quadratic non-residue in $\mathbb{F}_{p^3}$. Let $g = g_0 + g_1\sigma \in \mathbb{F}^*_{p^6}$. Then $g^2 = (g_0^2 + g_1^2c) + 2g_0g_1\sigma$ can be computed at a cost dominated by 2 multiplications in $\mathbb{F}_{p^3}$ because $g_0^2 + g_1^2c = (g_0 + g_1c)(g_0 + g_1) - (c+1)g_0g_1$ and we may again ignore the cost of addition, subtraction, and multiplication by c.

SL-squaring: Let $p \equiv 2, 5 \pmod 9$ and $g \in G_{\Phi_6(p)} \subset \mathbb{F}^*_{p^6}$. Using the algebraic relations induced by $g^{p^3+1} = g^{p^2-p+1} = 1$ on the coefficients of the vector representation of g over $\mathbb{F}_p$, Stam and Lenstra [28] showed that g^2 can be computed at a cost dominated by 6 multiplications in $\mathbb{F}_p$. Moreover, when $p \equiv 2 \pmod 3$ or $p \equiv 3 \pmod 4$, the algebraic relations induced by $g^{p^3+1} = 1$ on the coefficients of the vector representation of g over $\mathbb{F}_{p^3}$ were used to compute g^2 at a cost dominated by 2 squarings in $\mathbb{F}_{p^3}$. We call the Stam-Lenstra methods SL1-squaring and SL2-squaring, respectively.

GPS-squaring: Let $p \equiv 1 \pmod{12}$ and $g \in G_{\Phi_6(p)} \subset \mathbb{F}_{p^6}^*$. Using the algebraic relations induced by $g^{p^3+1} = g^{p^2-p+1} = 1$ on the coefficients of the vector representation of g over $\mathbb{F}_p$, Granger, Page and Stam [10] showed that g^2 can be computed at a cost dominated by 3 multiplications and 6 squarings in $\mathbb{F}_p$.

GS-squaring: Let $q \equiv 1 \pmod{6}$ be a prime power and $g \in G_{\Phi_6(q)} \subset \mathbb{F}_{q^6}^*$. Granger and Scott [12], using the algebraic relations induced by $g^{q^3+1} = g^{q^2-q+1} = 1$ on the coefficients of the vector representation of g over $\mathbb{F}_{q^2}$, showed that g^2 can be computed at a cost dominated by 3 squarings in $\mathbb{F}_{q^2}$.

Table 1 summarizes the dominating costs of the above-mentioned squaring algorithms. We let M_i and S_i denote multiplication and squaring costs in $\mathbb{F}_{p^i}$. We may assume *a)* $M_{3i} = 6M_i$ using Karatsuba's technique; *b)* $S_{2i} = 2M_i$ using general squaring as above; and *c)* $S_{3i} = M_i + 4S_i$ using Chung and Hasan's generalized Tom-Cook squaring formulae SQR_3 [6].

TABLE 1. A summary of squaring algorithms in $G_{\Phi_6(p)}$.

Algorithms	Cost	Restriction
Compressed representation		
LUC	$1S_3 = 1M_1 + 4S_1$	decompression
XTR	$1S_2 = 2M_1$	decompression
TB2	$2M_3 = 12M_1$	
Full representation		
General	$2M_3 = 12M_1$	
SL1	$6M_1$	$p \equiv 2, 5 \pmod{9}$
SL2	$2S_3 = 2M_1 + 8S_1$	$p \equiv 2 \pmod{3}, p \equiv 3 \pmod{4}$
GPS	$3M_1 + 6S_1$	$p \equiv 1 \pmod{12}$
GS	$3S_2 = 6M_1$	$p \equiv 1 \pmod{6}$

We should note that the squaring algorithms described above for $G_{\Phi_6(p)}$ can be generalized to obtain squaring algorithms for $G_{\Phi_{6i}(p)} \subset \mathbb{F}_{p^{6i}}^*$ when all prime divisors of i divide 6. Indeed, one can replace p by $q = p^i$ in the arguments and note that $G_{\Phi_{6i}(p)} = G_{\Phi_6(q)} \subset \mathbb{F}_{q^6}^*$.[1] The dominating costs of the corresponding squaring algorithms may be obtained by replacing each M_j and S_j in Table 1 by $M_{j \cdot i}$ and $S_{j \cdot i}$, respectively (and the restrictions on p in Table 1 should then be read as restrictions on q).

Even though XTR-squaring seems to be the fastest squaring algorithm for $G_{\Phi_{6i}(p)} \subset \mathbb{F}_{p^{6i}}^*$ (or, $G_{\Phi_6(q)} \subset \mathbb{F}_{q^6}^*$ with $q = p^i$) for $i \in \{1, 2, 3, 4\}$, it suffers from the non-multiplicative property of the trace function as mentioned in Section 1. Consequently, GS-squaring and SL1-squaring seem to be the best squaring algorithms that can be easily deployed in cryptographic algorithms. In addition, GS-squaring has the extra advantage that it allows the use of *pairing-friendly* or *towering-friendly* fields for more efficient implementation of pairing-based cryptographic protocols; see [17, 3].

[1] Under this generalization the GPS-squaring algorithm seems to scale better than the one described in [10] for $i = 2, 3$ and 4.

3. A new squaring formula in cyclotomic subgroups

Let $q = p^i \equiv 1 \pmod 6$ be a prime power and $G_{\Phi_6(q)} \subset \mathbb{F}_{q^6}^*$. We let $\mathbb{F}_{q^2} = \mathbb{F}_q(w)$ where $w^2 = c$ for some sextic non-residue $c \in \mathbb{F}_q$; and $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}(\sigma)$ where $\sigma^3 = w$. Then it is easy to show that

$$w^q = -w, \tag{3.1}$$

$$\sigma^q = m\sigma, \tag{3.2}$$

where $m \in \mathbb{F}_q$ is some primitive sixth root of unity.

If $g \in \mathbb{F}_{q^6}$ then we write

$$g = (g_0 + g_1 w) + (g_2 + g_3 w)\sigma + (g_4 + g_5 w)\sigma^2,$$

where $g_i \in \mathbb{F}_q$. In particular, if $g \in G_{\Phi_6(q)}$ then

$$g^{q^3+1} = g^{q^2-q+1} = 1 \tag{3.3}$$

and using (3.3) together with (3.1) and (3.2) we obtain the following nine relations for g_i's:

$$\begin{aligned}
P_1 &: \quad 2g_0g_4 - g_2^2 - (2g_1g_5 - g_3^2)c = 0,\\
P_2 &: \quad -2g_1g_2 + g_4^2 + 2g_0g_3 - g_5^2c = 0,\\
P_3 &: \quad g_0^2 - 1 - (g_1^2 + 2g_2g_5 - 2g_3g_4)c = 0,\\
P_4 &: \quad (1 - g_0)g_5 + 2g_2g_3 - g_1g_4 = 0,\\
P_5 &: \quad (-1 - g_0)g_4 + g_2^2 + (g_3^2 - g_1g_5)c = 0,\\
P_6 &: \quad (g_0 + 1)g_3 + g_1g_2 - g_4^2 - g_5^2c = 0,\\
P_7 &: \quad (g_0 - 1)g_2 + (g_3g_1 - 2g_5g_4)c = 0,\\
P_8 &: \quad (1 + 2g_0)g_1 - g_2g_4 - g_3g_5c = 0,\\
P_9 &: \quad g_0(g_0 - 1) + (g_1^2 - g_2g_5 - g_3g_4)c = 0.
\end{aligned}$$

Note that each relation P_i is independent of the choice of the primitive sixth root of unity m. Therefore, without loss of generality, we associate a 7-variate polynomial $P_i(X) = P_i(x_0, \ldots, x_5, y)$ to each relation P_i above. That is, if $g = (g_0 + g_1 w) + (g_2 + g_3 w)\sigma + (g_4 + g_5 w)\sigma^2 \in G_{\Phi_6(q)}$ then $P_i(g_0, \ldots, g_5, c) = 0$ for all $i = 1, \ldots, 9$. Thus, every element in $G_{\Phi_6(q)}$ corresponds to an $\mathbb{F}_q$-point in the variety defined by the ideal $\langle P_1(X), \ldots, P_9(X)\rangle$.

In fact, we may think of each $P_i(X)$ as a 7-variate polynomial defined over the field of rational numbers $\mathbb{Q}$, and define an ideal $\mathcal{I} = \langle P_1(X), \ldots, P_9(X)\rangle$ over $\mathbb{Q}$. Next, we compute a Groebner basis over $\mathbb{Q}$ of $\mathcal{I}$ with respect to some fixed ordering on the set of monomials, which in turn yields a factor-3/2 compression function $\mathcal{C}$ for elements $g \in G_{\Phi_6(q)}$ with the property that given $\mathcal{C}(g)$, g can be recovered uniquely at a cost dominated by an inversion in $\mathbb{F}_q$. We describe the compression and decompression functions in the following theorem.

Theorem 3.1. *Let $q \equiv 1 \pmod 6$ be a prime power. Let $\mathbb{F}_{q^2} = \mathbb{F}_q(w)$ where $w^2 = c$ for some sextic non-residue $c \in \mathbb{F}_q$, and let $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}(\sigma)$ where $\sigma^3 = w$. Let $g = (g_0 + g_1 w) + (g_2 + g_3 w)\sigma + (g_4 + g_5 w)\sigma^2 \in G_{\Phi_6(q)} \setminus \{1\} \subset \mathbb{F}_{q^6}^*$. Define the compression function $\mathcal{C}$ and the decompression function $\mathcal{D}$ as follows*

$$\mathcal{C}(g) = [g_2, g_3, g_4, g_5],$$

$$\mathcal{D}([\tilde{g}_2, \tilde{g}_3, \tilde{g}_4, \tilde{g}_5]) \;=\; (\tilde{g}_0 + \tilde{g}_1 w) + (\tilde{g}_2 + \tilde{g}_3 w)\sigma + (\tilde{g}_4 + \tilde{g}_5 w)\sigma^2,$$

where

$$\begin{cases} \tilde{g}_1 = \frac{\tilde{g}_5^2 c + 3\tilde{g}_4^2 - 2\tilde{g}_3}{4\tilde{g}_2}, \quad \tilde{g}_0 = (2\tilde{g}_1^2 + \tilde{g}_2\tilde{g}_5 - 3\tilde{g}_3\tilde{g}_4)c + 1, & \text{if } \tilde{g}_2 \neq 0; \\ \tilde{g}_1 = \frac{2\tilde{g}_4\tilde{g}_5}{\tilde{g}_3}, \quad \tilde{g}_0 = (2\tilde{g}_1^2 - 3\tilde{g}_3\tilde{g}_4)c + 1, & \text{if } \tilde{g}_2 = 0. \end{cases}$$

Then $\mathcal{D}$ is well-defined for all $\mathcal{C}(g)$ with $g \in G_{\Phi_6(q)} \setminus \{1\}$, and $\mathcal{D}(\mathcal{C}(g)) = g$ for all $g \in G_{\Phi_6(q)} \setminus \{1\}$.

Proof. Let $g = (g_0 + g_1 w) + (g_2 + g_3 w)\sigma + (g_4 + g_5 w)\sigma^2 \in G_{\Phi_6(q)} \setminus \{1\}$. If $g_2 = 0$ and $g_3 = 0$ then one can verify using the relations P_i that $g_1 = g_4 = g_5 = 0$ and $g_0 = 1$. Therefore, g_2 and g_3 cannot both be zero proving that $\mathcal{D}$ is well-defined for all $\mathcal{C}(g)$ with $g \in G_{\Phi_6(q)} \setminus \{1\}$.

To show that $\mathcal{D}(\mathcal{C}(g)) = g$ for all $g \in G_{\Phi_6(q)} \setminus \{1\}$, we compute a Groebner basis over $\mathbb{Q}$ of the ideal $\mathcal{I} = \langle P_1(X), \ldots P_9(X) \rangle$ with respect to the lexicographical ordering of the monomials with $x_0 > x_1 > x_5 > x_4 > x_3 > x_2 > y$. It can be verified using Magma with the commands

$$\begin{aligned} &\mathrm{R} < x_0, x_1, x_5, x_4, x_3, x_2, y >:= \mathrm{PolynomialRing}(\mathrm{RationalField}(), 7); \\ &\mathrm{B1} := [\mathrm{R}!P_1, \mathrm{R}!P_2, \mathrm{R}!P_3, \mathrm{R}!P_4, \mathrm{R}!P_5, \mathrm{R}!P_6, \mathrm{R}!P_7, \mathrm{R}!P_8, \mathrm{R}!P_9]; \\ &\mathrm{I1} := \mathrm{ideal} < \mathrm{R} | \mathrm{B1} >; \\ &\mathrm{GB1} := \mathrm{GroebnerBasis}(\mathrm{I1}); \\ &\mathrm{B2} := [\mathrm{R}!P_1, \mathrm{R}!P_2, \mathrm{R}!P_3, \mathrm{R}!P_4, \mathrm{R}!P_5, \mathrm{R}!P_6, \mathrm{R}!P_7, \mathrm{R}!P_8, \mathrm{R}!P_9, \mathrm{R}!x_2]; \\ &\mathrm{I2} := \mathrm{ideal} < \mathrm{R} | \mathrm{B2} >; \\ &\mathrm{GB2} := \mathrm{GroebnerBasis}(\mathrm{I2}); \end{aligned}$$

that

$$x_0 - (2x_1^2 + x_2x_5 - 3x_3x_4)y - 1,$$
$$x_1x_2 + \frac{x_3}{2} - \frac{3x_4^2}{4} - \frac{x_5^2 y}{4}$$

are two polynomials in the basis GB1; and

$$x_1x_3 - 2x_4x_5$$

is a polynomial in the basis GB2. □

By Theorem 3.1, we know that if $g = (g_0+g_1w)+(g_2+g_3w)\sigma+(g_4+g_5w)\sigma^2 \in G_{\Phi_6(q)}$ then $\mathcal{C}(g) = [g_2, g_3, g_4, g_5]$ determines g uniquely. This suggests a squaring formula that uses the compressed representation of elements in $G_{\Phi_6(q)}$ which we present in Theorem 3.2.

Theorem 3.2. *Let $q \equiv 1 \pmod 6$ be a prime power. Let $\mathbb{F}_{q^2} = \mathbb{F}_q(w)$ where $w^2 = c$ for some sextic non-residue $c \in \mathbb{F}_q$, and let $\mathbb{F}_{q^6} = \mathbb{F}_{q^2}(\sigma)$ where $\sigma^3 = w$. Let $g = (g_0+g_1w)+(g_2+g_3w)\sigma+(g_4+g_5w)\sigma^2 \in G_{\Phi_6(q)} \subset \mathbb{F}_{q^6}^*$. Let $\mathcal{C}$ be the compression function defined in the statement of Theorem 3.1. Let $h = g^2$, where $h = (h_0+h_1w)+(h_2+h_3w)\sigma+(h_4+h_5w)\sigma^2$. Then*

$$\mathcal{C}(g^2) = [h_2, h_3, h_4, h_5],$$

where

$$\begin{aligned} h_2 &= 2(g_2 + 3cB_{4,5}), \\ h_3 &= 3(A_{4,5} - (c+1)B_{4,5}) - 2g_3, \end{aligned}$$

$$\begin{aligned} h_4 &= 3(A_{2,3} - (c+1)B_{2,3}) - 2g_4, \\ h_5 &= 2(g_5 + 3B_{2,3}), \\ A_{i,j} &= (g_i + g_j)(g_i + cg_j), \\ B_{i,j} &= g_i g_j. \end{aligned}$$

Proof. Using $w^2 = c$ and $\sigma^3 = w$, we find that

$$\begin{aligned} h_0 &= g_0^2 + (g_1^2 + 2g_3g_4 + 2g_2g_5)c, \\ h_1 &= 2(g_0g_1 + g_2g_4 + g_3g_5c), \\ h_2 &= 2(g_0g_2 + (g_4g_5 + g_1g_3)c), \\ h_3 &= 2(g_0g_3 + g_1g_2) + g_4^2 + g_5^2c, \\ h_4 &= g_2^2 + 2g_0g_4 + (g_3^2 + 2g_1g_5)c, \\ h_5 &= 2(g_0g_5 + g_2g_3 + g_1g_4). \end{aligned}$$

Under the correspondence $x_i \leftrightarrow g_i$ and $y \leftrightarrow c$, we define the 7-variate polynomials $h_i(X)$. Next, we compute representatives of $h_i(X)$'s for $i \in \{2,3,4,5\}$ in the quotient ring $R/\mathcal{I}$, where $R = \mathbb{Q}[x_0, \ldots, x_5, y]$ with respect to the lexicographical ordering of monomials with $x_0 > x_1 > x_5 > x_4 > x_3 > x_2 > y$, and where $\mathcal{I} = \langle P_1(X), \ldots, P_9(X) \rangle$ is the ideal over $\mathbb{Q}$ defined earlier. It can be verified using Magma with the commands

R $< x_0, x_1, x_5, x_4, x_3, x_2, y >$:= PolynomialRing(RationalField(), 7);
B := [R!P_1, R!P_2, R!P_3, R!P_4, R!P_5, R!P_6, R!P_7, R!P_8, R!P_9];
I := ideal < R|B >;
R2 := R/I;
R2!h_2; R2!h_3; R2!h_4; R2!h_5;

that

$$\begin{aligned} h_2 &= 2(g_2 + 3g_4g_5c) = 2(g_2 + 3cB_{4,5}), \\ h_3 &= 3(g_4^2 + g_5^2c) - 2g_3 = 3(A_{4,5} - (c+1)B_{4,5}) - 2g_3, \\ h_4 &= 3(g_2^2 + g_3^2c) - 2g_4 = 3(A_{2,3} - (c+1)B_{2,3}) - 2g_4, \\ h_5 &= 2(g_5 + 3g_2g_3) = 2(g_5 + 3B_{2,3}) \end{aligned}$$

in the quotient ring $R/\mathcal{I}$.

□

Corollary 3.3. *Let $q \equiv 1 \pmod 6$ and $g \in G_{\Phi_6(q)} \subset \mathbb{F}_{q^6}^*$. Then $\mathcal{C}(g^2)$ can be computed at a cost dominated by 4 multiplications in $\mathbb{F}_q$.*

4. Applications

The squaring formula in Theorem 3.2 works with a compressed representation of elements in $G_{\Phi_6(q)}$. The decompression cost for this representation is far less than that of other compressed-squaring algorithms such as XTR-squaring (LUC-squaring) where decompression is performed by finding a root of a third (second) degree irreducible polynomial defined over $\mathbb{F}_{q^2}$ ($\mathbb{F}_{q^3}$). When the full representation of a compressed element $\mathcal{C}(g)$ is needed (for example, in the multiplication step of square-and-multiply type exponentiation algorithms), the decompression function $\mathcal{D}$ can recover g uniquely at a cost

dominated by an inversion in $\mathbb{F}_q$. Consequently, comparing Corollary 3.3 with Table 1, our compressed-squaring algorithm together with the decompression function yields an exponentiation algorithm which is especially fast when the exponent has low Hamming weight.

4.1. **Exponentiation in $G_{\Phi_6(q)}$.** Let $q = p^i \equiv 1 \pmod 6$ and $g \in G_{\Phi_6(q)} \subset \mathbb{F}^*_{q^6}$. We present an exponentiation algorithm that uses the squaring formula in Theorem 3.2.

Let e be an ℓ-bit exponent with binary representation

$$e = e_{\ell-1}e_{\ell-2}\dots e_2e_1e_0,$$

where $e_{\ell-1} = 1$. Let $H_e = \{i : \ 1 \le i \le \ell - 1 \text{ and } e_i = 1\}$. Then

$$g^e = \prod_{i=0}^{\ell-1} g^{2^i} = g^{e_0} \prod_{i \in H_e} \mathcal{D}(\mathcal{C}(g^{2^i})).$$

Now, if

$$g^{2^i} = (g_{i,0} + g_{i,1}w) + (g_{i,2} + g_{i,3}w)\sigma + (g_{i,4} + g_{i,5}w)\sigma^2$$

then by Theorem 3.1

$$\mathcal{C}(g^{2^i}) = [g_{i,2}, g_{i,3}, g_{i,4}, g_{i,5}]$$

and, assuming without loss of generality that $g_{i,2} \neq 0$,

$$\mathcal{D}(\mathcal{C}(g^{2^i})) = (x_i + \frac{y_i}{z_i}w) + (g_{i,2} + g_{i,3}w)\sigma + (g_{i,4} + g_{i,5}w)\sigma^2,$$

where

$$x_i = (2g_{i,1}^2 + g_{i,2}g_{i,5} - 3g_{i,3}g_{i,4})c + 1, \ y_i = g_{i,5}^2 c + 3g_{i,4}^2 - 2g_{i,3}, \quad \text{and } z_i = 4g_{i,2}.$$

Hence, g^e can be computed as follows.

(1) Compute $\mathcal{C}(g^{2^i})$ for $1 \le i \le \ell - 1$ with $(\ell - 1)$ successive squarings using Theorem 3.2 and store $\mathcal{C}(g^{2^i})$ for each $i \in H_e$.
(2) Compute and store (x_i, y_i, z_i) for each $i \in H_e$.
(3) Compute y_i/z_i for each $i \in H_e$.
(4) Compute $g^e = g^{e_0} \prod_{i \in H_e} \mathcal{D}(\mathcal{C}(g^{2^i}))$.

Now, let $|H_e| = N$, and let M_i, S_i and I_i denote multiplication, squaring and inversion costs in $\mathbb{F}_q = \mathbb{F}_{p^i}$, respectively. By Corollary 3.3, step (1) in the above algorithm has cost dominated by $(4(\ell-1))M_i$ and requires storage of $4N$ $\mathbb{F}_q$-elements. Using Montgomery's simultaneous inversion trick [20, 13], steps (2) and (3) have cost dominated by $N((1S_i + 2M_i) + 2S_i) + 3(N-1)M_i + 1I_i + (N)M_i$ and storage of $3N + 1$ $\mathbb{F}_q$-elements. Finally, step (4) can be computed at a cost of $(N)M_{6i}$.

Corollary 4.1. *Let $q = p^i \equiv 1 \pmod 6$ and $g \in G_{\Phi_6(q)} \subset \mathbb{F}^*_{q^6}$. Let e be an ℓ-bit exponent. Let $H_e = \{i : \ 1 \le i \le \ell - 1$ and $e_i = 1\}$, and let $|H_e| = N$. Then, g^e can be computed at a cost dominated by*

$$(4(\ell - 1))M_i + (6N - 3)M_i + (N)M_{6i} + (3N)S_i \ + 1I_i,$$

with a storage of $7N + 1$ $\mathbb{F}_q$-elements.

Note that the cost of exponentiation using GS-squaring (see [12] or Section 2) in the same setting as in Corollary 4.1 would be dominated by

$$6(\ell-1)M_i + (N)M_{6i}. \tag{4.1}$$

Hence, by Corollary 4.1, we would expect a 33% speed-up over GS-exponentiation as $N/\ell \to 0$.

4.2. **Speeding up pairing computations.** Let $E : y^2 = x^3 + b$ be a curve in the Barreto-Naehrig (BN) family of pairing-friendly curves with embedding degree $k = 12$ [2]. Then E is defined over $\mathbb{F}_p$ with $|E(\mathbb{F}_p)| = r$, where the primes p and r are parametrized as follows

$$\begin{aligned} p(u) &= 36u^4 + 36u^3 + 24u^2 + 6u + 1, \\ r(u) &= 36u^4 + 36u^3 + 18u^2 + 6u + 1. \end{aligned}$$

In general, a pairing computation on E is performed in two steps: Miller loop and final exponentiation. Scott [24] showed that the final exponentiation in the pairing computation can be done in two parts. In the first part, an element in $\mathbb{F}^*_{p^{12}}$ is raised to the power $(p^6-1)(p^2+1)$. This is the so-called *easy part* and requires a small number of multiplications, pth powerings and a single inversion, and yields an element $g \in G_{\Phi_6(p^2)} = G_{\Phi_{12}(p)} \subset \mathbb{F}^*_{p^{12}}$. In the second part of the final exponentiation, the so-called *hard part*, g is raised to the power

$$\Phi_{12}(p)/r = (p^4 - p^2 + 1)/r = \lambda_3 p^3 + \lambda_2 p^2 + \lambda_1 p + \lambda_0,$$

where

$$\begin{aligned} \lambda_3(u) &= 1, \\ \lambda_2(u) &= 6u^2 + 1, \\ \lambda_1(u) &= -36u^3 - 18u^2 - 12u + 1, \\ \lambda_0(u) &= -36u^3 - 30u^2 - 18u - 2. \end{aligned}$$

Note that when the BN parameter u is chosen to have low Hamming weight, one can first compute g^u, $g^{u^2} = (g^u)^u$ and $g^{u^3} = (g^{u^2})^u$ to minimize the number of multiplications in the hard part of the final exponentiation.

To be more concrete, one can choose $u = -(2^{62} + 2^{55} + 1)$ to obtain a BN-curve E defined over a 254-bit prime p with 254-bit prime order group $E(\mathbb{F}_p)$ [21]. In our operation counts, we will assume *a*) $M_2 = 3M_1$ and $M_{12} = 54M_1$ using Karatsuba's technique; *b*) $S_2 = 2M_1$ (see general squaring in Section 2.2); *c*) $I_2 = 1I_1 + 2M_1 + 2S_1$ [19]; and *d*) $M_1 = S_1$ and $I_1 = 50M_1$ (see [4, Section 3.1]).

GS-exponentiation seems to be the fastest of the previously-known methods to compute the hard part of the final exponentiation in the pairing computation. Using this method, each of g^{-u}, g^{-u^2} and g^{-u^3} can be computed at a cost dominated by (see (4.1))

$$6(\ell-1)M_i + (N)M_{6i} = 1224M_1, \tag{4.2}$$

where $\ell = 63$ and $N = 2$.

Now, we describe an exponentiation algorithm similar to the one in Section 4.1 but avoiding the storage requirements. Instead of performing simultaneous inversion, we begin the algorithm by computing compressed-squarings and perform a one-time inversion

(decompression) when a multiplication is required. Then we switch to GS-squaring. In summary, given g and $-u = 2^{62} + 2^{55} + 1$, g^{-u} can be computed as follows.

(1) Compute $\mathcal{C}(g^{2^{55}})$ with 55 successive squarings using Theorem 3.2.
(2) Decompress $\mathcal{C}(g^{2^{55}})$ to obtain $g^{2^{55}} = \mathcal{D}(\mathcal{C}(g^{2^{55}}))$.
(3) Compute $g^{2^{62}}$ with 7 successive GS-squarings.
(4) Compute $g^{-u} = g \cdot g^{2^{55}} \cdot g^{2^{62}}$.

The cost of step (2) is dominated by

$$(3S_2 + 2M_2) + 1I_2 + 1M_2 = 6M_1 + 6M_1 + 54M_1 + 3M_1 = 69M_1.$$

Similar to our previous analysis, one can verify that the cost of the above hybrid exponentiation algorithm is dominated by

$$(55 \cdot 4)M_2 + (7 \cdot 6)M_2 + 69M_1 + 2M_{12} = 963M_1. \tag{4.3}$$

Comparing (4.2) and (4.3), we would expect around a 21% speed-up for computing g^u, g^{u^2} and g^{u^3} in the hard part of the final exponentiation. According to [4, Section 4.2 and Table 3], computing g^u, g^{u^2} and g^{u^3} take 79% of the time of the final exponentiation, and the final exponentiation takes 42% of the time of the whole pairing computation. Hence, with our new exponentiation algorithm, we would expect a 17% speed-up for the final exponentiation, and a 7% speed-up for the pairing computation.

Recently, Aranha et al. proposed and implemented a variant of the squaring algorithm in Theorem 3.2 for the pairing computation over a BN curve parametrized by $u = -(2^{62}+2^{55}+1)$ [1]. They reported overall 5%−7% speed-ups for the pairing computation.

5. Other formulae for squaring

In this section, we describe a general method for finding efficient squaring formulae in cyclotomic subgroups $G_{\Phi_6(q)}$. While rediscovering some of the previously-known squaring formulae such as LUC-squaring, XTR-squaring and GS-squaring, our method yields new squaring formulae which might be good alternatives to the one in Section 3.

We use the same notation as in Section 3. In particular, let

$$g = (g_0 + g_1 w) + (g_2 + g_3 w)\sigma + (g_4 + g_5 w)\sigma^2 \in G_{\Phi_6(q)}$$

and

$$h = g^2 = (h_0 + h_1 w) + (h_2 + h_3 w)\sigma + (h_4 + h_5 w)\sigma^2.$$

Let $I \subseteq \{0, 1, \ldots, 5\}$. In order to obtain squaring formulae similar one in Theorem 3.2 we first need a (compression) function $\mathcal{C}$ such that $\mathcal{C}(g)$ can be determined as a function of $V_I(g) = \{g_i : \ i \in I\}$. Second, we need a family $\mathcal{S}_I = \{S_i : \ i \in I\}$ of formulae to compute each $h_i \in V_I(h)$ as a function of $V_I(g)$, say $h_i = S_i(V_I(g))$ for each $i \in I$. Consequently, we might represent a squaring formula $\mathcal{F} : \ \mathcal{C}(g) \mapsto \mathcal{C}(g^2) = \mathcal{C}(h)$ by a tuple

$$\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}.$$

For example, in Theorem 3.2 we had

$$I = \{2, 3, 4, 5\}, \ \mathcal{C}(g) = [g_2, g_3, g_4, g_5], \ \mathcal{S}_I = \{S_2, S_3, S_4, S_5\},$$
$$h_2 = S_2(V_I(g)) = 2(g_2 + 3g_4 g_5 c),$$
$$h_3 = S_3(V_I(g)) = 3(g_4^2 + g_5^2 c) - 2g_3,$$

$$h_4 = S_4(V_I(g)) = 3(g_2^2 + g_3^2 c) - 2g_4,$$
$$h_5 = S_5(V_I(g)) = 2(g_5 + 3g_2 g_3).$$

Remark 5.1. In the representation of $\mathcal{F}$, it seems necessary to require that the inverse of $\mathcal{C}$ can be efficiently computed in order to get efficient multi-exponentiation algorithms based on $\mathcal{F}$. However, we will relax this condition for now.

We already know from the proof of Theorem 3.2 that if $h = g^2$ then

$$\begin{aligned}
h_0 &= g_0^2 + (g_1^2 + 2g_3 g_4 + 2g_2 g_5)c,\\
h_1 &= 2(g_0 g_1 + g_2 g_4 + g_3 g_5 c),\\
h_2 &= 2(g_0 g_2 + (g_4 g_5 + g_1 g_3)c),\\
h_3 &= 2(g_0 g_3 + g_1 g_2) + g_4^2 + g_5^2 c,\\
h_4 &= g_2^2 + 2g_0 g_4 + (g_3^2 + 2g_1 g_5)c,\\
h_5 &= 2(g_0 g_5 + g_2 g_3 + g_1 g_4).
\end{aligned}$$

In fact, the squaring formula in Theorem 3.2, or in other words $\mathcal{S}_I = \{S_2, S_3, S_4, S_5\}$, was found by computing representatives of $h_i(X)$'s for $i \in \{2,3,4,5\}$ in the quotient ring $R/\mathcal{I}$, where $R = \mathbb{Q}[x_0, \ldots, x_5, y]$ with respect to the lexicographical ordering of monomials with $x_0 > x_1 > x_5 > x_4 > x_3 > x_2 > y$, and where $\mathcal{I} = \langle P_1(X), \ldots, P_9(X)\rangle$.

In order to capture a wider class of squaring formulae $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$, we will compute representatives of $h_i(X)$ for $i \in \{0, \ldots, 5\}$ in the quotient ring $R/\mathcal{I}$ by varying over all the $7! = 5040$ orderings of the variables $\{x_0, \ldots, x_5, y\}$. To do so, we let o be some fixed ordering in the set $\mathcal{O}$ of all orderings of the variables $\{x_0, \ldots, x_5, y\}$ and denote by R_o the ring $\mathbb{Q}[x_0, \ldots, x_5, y]$ with respect to the lexicographical ordering of monomials with ordering o.

We define

$$\mathcal{H}_i = \{\bar{h}_{i,o} : \ \bar{h}_{i,o} = h_i \in R_o/\mathcal{I}, o \in \mathcal{O}\}.$$

Since each $\bar{h}_{i,o}$ defines a unique S_i (on some subset $V_I(g)$), we may replace, without loss of generality, $\bar{h}_{i,o}$'s by $S_{i,j}$'s in $\mathcal{H}_i$. We list $\mathcal{H}_i$ for $i = 0, 1, \ldots, 5$ in Appendix A.

Note that for any $I \subseteq \{0, 1, 2 \ldots, 5\}$, there is a squaring formula $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ only if for each $i \in I$ there is some $S_{i,j} \in \mathcal{H}_i$ that is defined on $V_I(g)$. From Appendix A, we deduce that $S_{0,j} \in \mathcal{H}_0$ is well defined on a domain $V_I(g)$ only if $V_I(g)$ contains one of the subsets in the *minimal domain set* D_0 of $\mathcal{H}_0$, where

$$\begin{aligned}
D_0 = \{&\{g_0, g_1\}, \{g_0, g_2, g_5\}, \{g_0, g_3, g_4\}, \{g_1, g_2, g_5\}, \{g_0, g_1, g_2, g_3\},\\
&\{g_0, g_1, g_3, g_4\}, \{g_0, g_1, g_4, g_5\}, \{g_0, g_2, g_3, g_4, g_5\}\}
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
D_1 = \{&\{g_0, g_1\}, \{g_1, g_2, g_5\}, \{g_1, g_2, g_3, g_4\}, \{g_1, g_2, g_3, g_5\},\\
&\{g_1, g_2, g_4, g_5\}, \{g_1, g_3, g_4, g_5\}, \{g_0, g_1, g_2, g_3, g_5\},\\
&\{g_0, g_1, g_2, g_4, g_5\}, \{g_1, g_2, g_3, g_4, g_5\}\},\\
D_2 = \{&\{g_2, g_4, g_5\}, \{g_0, g_1, g_2, g_3\}, \{g_1, g_2, g_3, g_5\}, \{g_1, g_2, g_3, g_4, g_5\}\},\\
D_3 = \{&\{g_0, g_3, g_4\}, \{g_0, g_3, g_5\}, \{g_1, g_2, g_5\}, \{g_3, g_4, g_5\},\\
&\{g_0, g_1, g_2, g_3\}, \{g_1, g_2, g_3, g_4\}, \{g_1, g_2, g_3, g_5\}\},\\
D_4 = \{&\{g_0, g_2, g_4\}, \{g_0, g_3, g_4\}, \{g_1, g_2, g_5\}, \{g_2, g_3, g_4\},
\end{aligned}$$

$$\begin{aligned} & \{g_0, g_1, g_4, g_5\}, \{g_1, g_2, g_4, g_5\}, \{g_1, g_3, g_4, g_5\}\}, \\ D_5 \;=\; & \{\{g_2, g_3, g_5\}, \{g_0, g_1, g_4, g_5\}, \{g_1, g_2, g_4, g_5\}, \{g_1, g_2, g_3, g_4, g_5\}\}. \end{aligned}$$

Now, from the minimal domain sets D_i's and $\mathcal{H}_i$'s in Appendix A we can write other squaring formulae $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ as follows.

5.1. **SQR$_{01}$: $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ with $|I| = 2$.**

$$\begin{aligned} I = \{0,1\},\ \mathcal{C}(g) = [g_0, g_1],\ \mathcal{S}_I = \{S_{0,3}, S_{1,1}\} \\ h_0 = S_{0,3} = 3g_0^2 - 2g_0 + 3g_1^2 c, \\ h_1 = S_{1,1} = 6g_0 g_1 + 2g_1. \end{aligned}$$

This formula is the only one with $|I| = 2$. Its cost is dominated by 2 multiplications in $\mathbb{F}_q$ as one can write

$$h_0 = 3((g_0 + g_1)(g_0 + g_1 c) - (c+1)g_0 g_1) - 2g_0.$$

In fact, this formula is a rediscovery of the XTR-squaring because

$$\mathrm{Tr}_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}(g) = 3(g_0 + g_1 w)$$

can be uniquely determined using $V_I(g) = \{g_0, g_1\}$. Note that the compression function $\mathcal{C}$ in this case cannot have an inverse as $\mathcal{C}(g) = \mathcal{C}(g^{q^2}) = \mathcal{C}(g^{q^4})$.

5.2. **SQR$_{034}$: $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ with $|I| = 3$.**

$$\begin{aligned} I = \{0,3,4\},\ \mathcal{C}(g) = [g_0, g_3, g_4],\ \mathcal{S}_I = \{S_{0,1}, S_{3,2}, S_{4,1}\} \\ h_0 = S_{0,1} = 2g_0^2 + 4g_3 g_4 c - 1, \\ h_3 = S_{3,2} = 4g_0 g_3 + 2g_4^2, \\ h_4 = S_{4,1} = 4g_0 g_4 + 2g_3^2 c. \end{aligned}$$

This formula is the only one with $|I| = 3$, and is a rediscovery of the LUC-squaring because

$$\mathrm{Tr}_{\mathbb{F}_{q^6}/\mathbb{F}_{q^3}}(g) = 2(g_0 + g_3 w\sigma + g_4 \sigma^2)$$

can be uniquely determined using $V_I(g) = \{g_0, g_3, g_4\}$. In fact, using

$$\mathrm{Tr}_{q^6,q^3}(g^2) = \mathrm{Tr}_{q^6,q^3}(g)^2 - 2$$

and the 3-way squaring formula in [6] one can show that

$$\begin{aligned} h_0 &= 2(T_0 + T_1 c) - 1, \\ h_3 &= (T_1 + T_2) - 2(T_0 + T_4), \\ h_4 &= 2(T_4 c - T_3) + (T_1 - T_2), \end{aligned}$$

where

$$\begin{aligned} T_0 &= g_0^2, \\ T_1 &= (g_0 + g_3 + g_4)^2, \\ T_2 &= (g_0 + g_3 - g_4)^2, \\ T_3 &= 2g_3 g_4, \\ T_4 &= g_3^2. \end{aligned}$$

Hence, the total cost is dominated by 1 squaring in $\mathbb{F}_{q^3}$, or by 4 squarings and 1 multiplication in $\mathbb{F}_q$. Note that the compression function $\mathcal{C}$ in this case cannot have an inverse as $\mathcal{C}(g) = \mathcal{C}(g^{q^3})$.

5.3. **$\mathbf{SQR}_{2345}$: $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ with $|I| = 4$.**

$$I = \{2,3,4,5\},\ \mathcal{C}(g) = [g_2, g_3, g_4, g_5],\ \mathcal{S}_I = \{S_{2,1}, S_{3,4}, S_{4,5}, S_{5,1}\}$$
$$h_2 = S_{2,1} = 2g_2 + 6g_4g_5c,$$
$$h_3 = S_{3,4} = 3g_4^2 + 3g_5^2c - 2g_3,$$
$$h_4 = S_{4,5} = 3g_2^2 + 3g_3^2c - 2g_4,$$
$$h_5 = S_{5,1} = 2g_5 + 6g_2g_3.$$

The above formula is a rediscovery of the squaring formula in Theorem 3.2, where we show that $\mathcal{C}(g^2)$ can be computed at a cost dominated by 4 multiplications in $\mathbb{F}_q$. Next, we observe that $\mathcal{C}(g^2)$ can also be computed at a cost dominated by 2 squarings in $\mathbb{F}_{q^2}$. This follows because, inspired by GS-squaring [12], we can write

$$h_2 + h_3w = S_{2,1} + S_{3,4}w = 3w(g_4 + g_5w)^2 + 2(g_2 - g_3w),$$
$$h_4 + h_5w = S_{4,5} + S_{5,2}w = 3(g_2 + g_3w)^2 - 2(g_4 - g_5w),$$

which requires 2 squarings in $\mathbb{F}_{q^2}$.

5.4. **$\mathbf{SQR}_{0134}$: $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ with $|I| = 4$.**

$$I = \{0,1,3,4\},\ \mathcal{C}(g) = [g_0, g_1, g_3, g_4],\ \mathcal{S}_I = \{S_{0,1}, S_{1,1}, S_{3,2}, S_{4,1}\}$$
$$h_0 = S_{0,1} = 2g_0^2 + 4g_3g_4c - 1,$$
$$h_1 = S_{1,1} = 6g_0g_1 + 2g_1,$$
$$h_3 = S_{3,2} = 4g_0g_3 + 2g_4^2,$$
$$h_4 = S_{4,1} = 4g_0g_4 + 2g_3^2c.$$

In this formula, one can compute $\{h_0, h_3, h_4\}$ at a cost dominated by 4 squarings and 1 multiplication in $\mathbb{F}_q$ (see Section 5.2). Hence, $\{h_0, h_1, h_3, h_4\}$ can be computed at a cost dominated by 4 squarings and 2 multiplications in $\mathbb{F}_q$. Similarly as in the proof of Theorem 3.1, we can show that an inverse to the compression function $\mathcal{C}$ can be given as follows.

$$\mathcal{D}([\tilde{g}_0, \tilde{g}_1, \tilde{g}_3, \tilde{g}_4]) = (\tilde{g}_0 + \tilde{g}_1w) + (\tilde{g}_2 + \tilde{g}_3w)\sigma + (\tilde{g}_4 + \tilde{g}_5w)\sigma^2,$$

where

$$\tilde{g}_2 = \frac{\tilde{g}_3(\tilde{g}_0 - 1) + 2\tilde{g}_4^2}{3\tilde{g}_1},$$
$$\tilde{g}_5 = \frac{\tilde{g}_4(\tilde{g}_0 - 1) + 2\tilde{g}_2^2c}{3\tilde{g}_1c}.$$

If $g \in G_{\Phi_6(q)} \setminus \{1\}$ then g_1 can never equal zero because

$$\mathrm{Tr}_{\mathbb{F}_{q^6}/\mathbb{F}_{q^2}}(g^2) = g_0 + g_1w \in \mathbb{F}_{q^2} \setminus \mathbb{F}_q.$$

Hence, the decompression function $\mathcal{D}$ is well defined and $\mathcal{D}(\mathcal{C}(g)) = g$ for all $g \in G_{\Phi_6(q)} \setminus \{1\}$. The decompression can be performed at a cost dominated by 4 multiplications, 2 squarings and 1 inversion in $\mathbb{F}_q$.

5.5. **Other formulae $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ with $|I| = 4$.** There are 4 more classes of squaring formulae with $|I| = 4$ and we list one from each class that seems to be the most efficient one in its class.

$$
\begin{aligned}
I &= \{0,1,2,3\},\ \mathcal{C}(g) = [g_0, g_1, g_2, g_3],\ \mathcal{S}_I = \{S_{0,3}, S_{1,1}, S_{2,2}, S_{3,3}\}\\
h_0 &= S_{0,3} = 3g_0^2 - 2g_0 + 3g_1^2 c,\\
h_1 &= S_{1,1} = 6g_0 g_1 + 2g_1,\\
h_2 &= S_{2,2} = 3g_0 g_2 + 3g_1 g_3 c - g_2,\\
h_3 &= S_{3,3} = 3g_0 g_3 + 3g_1 g_2 + g_3.
\end{aligned}
$$

$$
\begin{aligned}
I &= \{0,1,4,5\},\ \mathcal{C}(g) = [g_0, g_1, g_4, g_5],\ \mathcal{S}_I = \{S_{0,3}, S_{1,1}, S_{4,3}, S_{5,2}\}\\
h_0 &= S_{0,3} = 3g_0^2 + 3g_1^2 c - 2g_0,\\
h_1 &= S_{1,1} = 6g_0 g_1 + 2g_1,\\
h_4 &= S_{4,3} = 3g_0 g_4 + 3g_1 g_5 c + g_4,\\
h_5 &= S_{5,2} = 3g_0 g_5 + 3g_1 g_4 - g_5.
\end{aligned}
$$

$$
\begin{aligned}
I &= \{1,2,3,5\},\ \mathcal{C}(g) = [g_1, g_2, g_3, g_5],\ \mathcal{S}_I = \{S_{1,5}, S_{2,1}, S_{3,1}, S_{5,1}\}\\
h_1 &= S_{1,5} = 12g_5^3 c^2 + (12g_1^3 + 12g_1 g_2 g_5 - 36g_2 g_3^2 - 24g_3 g_5)c + 8g_1,\\
h_2 &= S_{2,1} = 2g_2 + 6g_4 g_5 c,\\
h_3 &= S_{3,1} = 4g_1 g_2 + 2g_5^2 c,\\
h_5 &= S_{5,1} = 6g_2 g_3 + 2g_5.
\end{aligned}
$$

$$
\begin{aligned}
I &= \{1,2,4,5\},\ \mathcal{C}(g) = [g_1, g_2, g_4, g_5],\ \mathcal{S}_I = \{S_{1,3}, S_{2,1}, S_{4,2}, S_{5,3}\}\\
h_1 &= S_{1,3} = (12g_1 g_2 g_5 + 12g_1^3 - 36g_4^2 g_5)c - 24g_2 g_4 + 8g_1 + 12g_2^3,\\
h_2 &= S_{2,1} = 2g_2 + 6g_4 g_5 c,\\
h_4 &= S_{4,2} = 4g_1 g_5 c + 2g_2^2,\\
h_5 &= S_{5,3} = -12g_1 g_2^2 + 9g_2 g_4^2 + 3g_2 g_5^2 c + 2g_5.
\end{aligned}
$$

For each of the above 4 formulae, one can write a decompression function $\mathcal{D}$ such that $\mathcal{D}$ is well defined and $\mathcal{D}(\mathcal{C}(g)) = g$ for all $g \in G_{\Phi_6(q)}$. Our analysis shows that in the first two squaring formulae, the decompression functions require 1 inversion in $\mathbb{F}_q$, and computing $\mathcal{C}(g^2)$ requires more than 4 multiplications in $\mathbb{F}_q$. Therefore, they do not seem to yield better algorithms than the squaring formula in Theorem 3.2. In the latter two squaring formulae, the decompression functions do not require an inversion, and they can be given as follows:

$$\mathcal{D}([g_1, g_2, g_3, g_5]) = (g_0 + g_1 w) + (g_2 + g_3 w)\sigma + (g_4 + g_5 w)\sigma^2,$$

where

$$
\begin{aligned}
g_4 &= 1/2(g_2^2 + 3g_3^2 c) - 2g_1 g_5 c,\\
g_0 &= (2g_1^2 + g_2 g_5 - 3g_3 g_4)c + 1,
\end{aligned}
$$

and

$$\mathcal{D}([g_1, g_2, g_4, g_5]) = (g_0 + g_1 w) + (g_2 + g_3 w)\sigma + (g_4 + g_5 w)\sigma^2,$$

where

$$g_3 = 1/2(g_5^2 c + 3g_4^2) - 2g_1 g_2,$$
$$g_0 = (2g_1^2 + g_2 g_5 - 3g_3 g_4)c + 1,$$

respectively. The above formulae yield factor-3/2 compression for elements $g \in G_{\Phi_6(q)}$ with the property that decompression can be performed at a cost dominated by 3 multiplications and 3 squarings in $\mathbb{F}_q$. However, computing $\mathcal{C}(g^2)$ requires more than 6 multiplications in $\mathbb{F}_q$. Therefore, they do not seem to yield better algorithms than GS-squaring.

5.6. **Squaring formulae $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ with $|I| = 5$.** There are 6 classes of squaring formulae with $|I| = 5$. Our analysis shows in all of these formulae, computing $\mathcal{C}(g^2)$ requires more than 4 multiplications in $\mathbb{F}_q$. Therefore, there is no hope that they would yield better exponentiation algorithms than the formula in Theorem 3.2 unless decompression can be achieved without doing an inversion in $\mathbb{F}_q$ (recall that the decompression function of the squaring formula in Theorem 3.2 requires an inversion in $\mathbb{F}_q$). We found one such formula, where $\mathcal{C}(g^2)$ can be computed at a cost dominated by 5 multiplications in $\mathbb{F}_q$, and the decompression can be performed at a cost dominated by 2 multiplications and 1 squaring in $\mathbb{F}_q$.

The formula can be seen as an extension of the squaring formula in Theorem 3.2 and it is given as follows.

$$I = \{1,2,3,4,5\},\ \mathcal{C}(g) = [g_0, g_2, g_3, g_4, g_5],$$
$$\mathcal{S}_I = \{S_{1,2}, S_{2,1}, S_{3,4}, S_{4,5}, S_{5,1}\},$$
$$h_1 = -g_1 + 3(C_{2,3,4,5} - B_{4,5} - B_{2,3}c),$$
$$h_2 = 2(g_2 + 3cB_{4,5}),$$
$$h_3 = 3(A_{4,5} - (c+1)B_{4,5}) - 2g_3,$$
$$h_4 = 3(A_{2,3} - (c+1)B_{2,3}) - 2g_4,$$
$$h_5 = 2(g_5 + 3B_{2,3}),$$

where

$$A_{i,j} = (g_i + g_j)(g_i + cg_j),$$
$$B_{i,j} = g_i g_j,$$
$$C_{2,3,4,5} = (g_2 + g_5)(g_3 c + g_4).$$

It is clear from Theorem 3.1 that an inverse to the compression function $\mathcal{C}$ can be given as follows.

$$\tilde{g}_0 = (2\tilde{g}_1^2 + \tilde{g}_2\tilde{g}_5 - 3\tilde{g}_3\tilde{g}_4)c + 1.$$

There is one class of squaring formulae with $|I| = 6$ and we list below two formulae that are the most efficient ones according to our analysis; the first can be seen as an extension of the squaring formula in Theorem 3.2, and the second one can be seen as an extension of the squaring formula in Section 5.4.

5.7. **A squaring formula $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ with $|I| = 6$.**

$$
\begin{aligned}
I &= \{0,1,2,3,4,5\},\ \mathcal{C}(g) = [g_0, g_1, g_2, g_3, g_4, g_5],\\
\mathcal{S}_I &= \{S_{0,3}, S_{1,1}, S_{2,1}, S_{3,4}, S_{4,5}, S_{5,1}\},\\
h_0 &= S_{0,3} = 3g_0^2 - 2g_0 + 3g_1^2 = 3(A_{0,1} - (c+1)B_{0,1}) - 2g_0,\\
h_1 &= S_{1,1} = 6g_0g_1 + 2g_1 = 2(g_1 + 3B_{0,1}),\\
h_2 &= S_{2,1} = 2g_2 + 6g_4g_5 = 2(g_2 + 3cB_{4,5}),\\
h_3 &= S_{3,4} = 3g_4^2 + 3g_5^2c - 2g_3 = 3(A_{4,5} - (c+1)B_{4,5}) - 2g_3,\\
h_4 &= S_{4,5} = 3g_2^2 + 3g_3^2c - 2g_4 = 3(A_{2,3} - (c+1)B_{2,3}) - 2g_4,\\
h_5 &= S_{5,1} = 2g_5 + 6g_2g_3 = 2(g_5 + 3B_{2,3}),
\end{aligned}
$$

where

$$
\begin{aligned}
A_{i,j} &= (g_i + g_j)(g_i + cg_j),\\
B_{i,j} &= g_ig_j.
\end{aligned}
$$

Note that $\mathcal{C}(g^2)$ can be computed at a cost dominated by 6 multiplications in $\mathbb{F}_q$, and there is no need of a decompression function. In fact, a closer look at the formulae [12] shows that the above formula is a rediscovery of GS-squaring.

5.8. **$\mathbf{SQR}_{012345}$: $\mathcal{F} = \{I, \mathcal{C}, \mathcal{S}_I\}$ with $|I| = 6$.**

$$
\begin{aligned}
I &= \{0,1,2,3,4,5\},\ \mathcal{C}(g) = [g_0, g_1, g_2, g_3, g_4, g_5],\\
\mathcal{S}_I &= \{S_{0,1}, S_{1,1}, S_{2,1}, S_{3,2}, S_{4,1}, S_{5,1}\},\\
h_0 &= S_{0,1} = 2g_0^2 + 4g_3g_4c - 1 = 2(T_0 + T_1c) - 1,\\
h_1 &= S_{1,1} = 6g_0g_1 + 2g_1 = 2(g_1 + 3B_{0,1}),\\
h_2 &= S_{2,1} = 2g_2 + 6g_4g_5c = 2(g_2 + 3cB_{4,5}),\\
h_3 &= S_{3,2} = 4g_0g_3 + 2g_4^2 = (T_1 + T_2) - 2(T_0 + T_4),\\
h_4 &= S_{4,1} = 4g_0g_4 + 2g_3^2c = 2(T_4c - T_3) + (T_1 - T_2),\\
h_5 &= S_{5,1} = 2g_5 + 6g_2g_3 = 2(g_5 + 3B_{2,3}),
\end{aligned}
$$

where

$$
\begin{aligned}
T_0 &= g_0^2,\\
T_1 &= (g_0 + g_3 + g_4)^2,\\
T_2 &= (g_0 + g_3 - g_4)^2,\\
T_3 &= 2g_3g_4,\\
T_4 &= g_3^2,\\
B_{i,j} &= g_ig_j.
\end{aligned}
$$

Using the above formula, $\mathcal{C}(g^2)$ can be computed at a cost dominated by 4 multiplications and 4 squarings in $\mathbb{F}_q$, and and there is no need of a decompression function.

6. Comparisons

In Tables 2, 3 and 4, we compare the most efficient squaring formulae in this paper with the squaring formula in [12] which is the fastest previously-known squaring formula that can be easily adapted for multi-exponentiation algorithms. We denote our squaring formula in Theorem 3.2 by SQR_{2345} (also see Section 5.3) as it can be written as a function of g_2, g_3, g_4, g_5. Similarly, we denote the squaring formulae in Sections 5.4, 5.6 and 5.8 by SQR_{0134}, SQR_{12345} and SQR_{012345}, respectively.

As before, we let M_i and S_i denote multiplication and squaring costs in $\mathbb{F}_{p^i}$, and assume *a)* $M_{2i} = 3M_i$, $M_{3i} = 6M_i$; and *b)* $S_1 = M_1$, $S_{2i} = 2M_i$, $S_{3i} = M_i + 4S_i$.

Table 2. A comparison of squaring algorithms in $G_{\Phi_6(q)} \subset \mathbb{F}^*_{q^6}$, where $q = p^i \equiv 1 \pmod 6$ and $i = 2^a 3^b$ with $a > 0$.

Algorithm	Squaring cost	Decompression cost
SQR_{2345}	$(6^b \cdot 3^a \cdot 4)M_1$	$1I_i + (6^b \cdot 3^a \cdot 5)M_1$
SQR_{0134}	$(6^b \cdot 3^a \cdot 14/3)M_1$	$1I_i + (6^b \cdot 3^a \cdot 16/3)M_1$
SQR_{12345}	$(6^b \cdot 3^a \cdot 5)M_1$	$(6^b \cdot 3^a \cdot 8/3)M_1$
SQR_{012345}	$(6^b \cdot 3^a \cdot 20/3)M_1$	0
GS-squaring [12]	$(6^b \cdot 3^a \cdot 6)M_1$	0

According to Table 2, SQR_{2345} is the fastest squaring algorithm. In particular, SQR_{2345} is 33% faster than GS-squaring. When evaluating the costs of squaring algorithms in Table 2, we assumed that $M_{2^a 3^b} = (6^b \cdot 3^a)M_1$ and $S_{2^a 3^b} = 2M_{2^{a-1}3^b} = (6^b \cdot 3^a \cdot 2/3)M_1$. In particular, the costs of SQR_{2345} and GS-squaring are computed as $2S_{2^{a+1}3^b} = 4M_{2^a 3^b} = (6^b \cdot 3^a \cdot 4)M_1$ and $3S_{2^{a+1}3^b} = 6M_{2^a 3^b} = (6^b \cdot 3^a \cdot 6)M_1$, respectively.

It is possible to obtain better (asymptotic) running time estimates for the algorithms listed in Table 2 because using $S_{3i} = M_i + 4S_i$ repetitively, we can show that

$$S_{2^a 3^b} = 6^b \cdot 3^a \cdot 1/2 \cdot (1 + (2^b/3^{b+1}))M_1 \text{ for } a, b > 0.$$

Note that $(M_{2^a 3^b}/S_{2^a 3^b}) \to 2$ as $b \to \infty$. Then, for example, the costs of SQR_{2345} and GS-squaring can be estimated as $2S_{2^{a+1}3^b} = (6^b \cdot 3^a \cdot 3)M_1$ and $3S_{2^{a+1}3^b} = (6^b \cdot 3^a \cdot 9/2)M_1$, respectively. We present an asymptotic comparison of the squaring algorithms in Table 3. According to Table 3, SQR_{2345} is the fastest squaring algorithm, and is 33% faster than GS-squaring.

Table 3. An asymptotic comparison of squaring algorithms in $G_{\Phi_6(q)} \subset \mathbb{F}^*_{q^6}$, where $q = p^i \equiv 1 \pmod 6$ and $i = 2^a 3^b$ with $a > 0$ and $b \to \infty$.

Algorithm	Squaring cost	Decompression cost
SQR_{2345}	$(6^b \cdot 3^a \cdot 3)M_1$	$1I_i + (6^b \cdot 3^a \cdot 9/2)M_1$
SQR_{0134}	$(6^b \cdot 3^a \cdot 4)M_1$	$1I_i + (6^b \cdot 3^a \cdot 5)M_1$
SQR_{12345}	$(6^b \cdot 3^a \cdot 5)M_1$	$(6^b \cdot 3^a \cdot 5/2)M_1$
SQR_{012345}	$(6^b \cdot 3^a \cdot 6)M_1$	0
GS-squaring [12]	$(6^b \cdot 3^a \cdot 9/2)M_1$	0

As we noted in Section 4, the decompression costs must be considered when adapting our squaring formulae in exponentiation algorithms. However, this does not seem to be a big issue when the exponent has low Hamming weight. For example, in Section 4.1

we discuss how SQR_{2345} can be used to improve the efficiency of exponentiation in $G_{\Phi_{12}(p)} \subset \mathbb{F}^*_{p^{12}}$ and the BN pairing computations. Similarly, from Table 4 we see that SQR_{2345} can be used to improve the efficiency of exponentiation in $G_{\Phi_k(p)} \subset \mathbb{F}^*_{p^k}$ and pairing computations that use elliptic curves with embedding degree k, for $k = 18, 24$.[2]

TABLE 4. A comparison of squaring algorithms in $G_{\Phi_6(q)} \subset \mathbb{F}^*_{q^6}$, $q \equiv 1 \pmod 6$.

$q = p^i$	Squaring cost	Decompression cost	Squaring cost	Decompression cost
	SQR_{2345}		SQR_{0134}	
i	$\min(4M_i, 2S_{2i})$	$1I_i + 3M_i + 3S_i$	$2M_i + 4S_i$	$1I_i + 4M_i + 2S_i$
$i = 1$	$4M_1$	$1I_1 + 6M_1$	$6M_1$	$1I_1 + 6M_1$
$i = 2$	$12M_1$	$1I_2 + 15M_1$	$14M_1$	$1I_2 + 16M_1$
$i = 3$	$22M_1$	$1I_3 + 33M_1$	$32M_1$	$1I_3 + 34M_1$
$i = 4$	$36M_1$	$1I_4 + 45M_1$	$42M_1$	$1I_4 + 48M_1$
	SQR_{12345}		SQR_{012345}	
i	$5M_i$	$2M_i + 1S_i$	$4M_i + 4S_i$	0
$i = 1$	$5M_1$	$3M_1$	$8M_1$	0
$i = 2$	$15M_1$	$8M_1$	$20M_1$	0
$i = 3$	$30M_1$	$17M_1$	$44M_1$	0
$i = 4$	$45M_1$	$24M_1$	$60M_1$	0
	GS-squaring [12]			
i	$\min(6M_i, 3S_{2i})$	0		
$i = 1$	$6M_1$	0		
$i = 2$	$18M_1$	0		
$i = 3$	$33M_1$	0		
$i = 4$	$54M_1$	0		

When the exponent is chosen at random in an exponentiation algorithm, it seems more advantageous to use one of SQR_{12345} or GS-squaring because decompression is a relatively cheap operation. Let e be an exponent with ℓ-bits. The width-w NAF representation of e contains on average $\ell/(w+1)$ nonzero digits which determines the number of multiplications and the number of decompression operations in the exponentiation algorithm. Then, the cost of computing $g^e \in G_{\Phi_6(p^i)} \subset \mathbb{F}^*_{p^{6i}}$ using SQR_{12345} and GS-squaring would be dominated by

$$\mathrm{EXP}_{12345}(i, w, \ell) = (5\ell)M_i + \frac{\ell}{w+1}(M_{6i}) + \frac{\ell}{w+1}(2M_i + 1S_i),$$
$$\mathrm{EXP}_{\mathrm{GS}}(i, w, \ell) = (3\ell)(S_{2i}) + \frac{\ell}{w+1}M_{6i},$$

respectively. Table 5 compares the exponentiation costs per bit of ℓ for particular cases of i and w.

7. Concluding remarks

We proposed new squaring formulae for cyclotomic subgroups $G_{\Phi_6(p)} \subset \mathbb{F}^*_{q^6}$, where $q \equiv 1 \pmod 6$, and demonstrated that the formulae can be used to speed up cryptographic protocols. Our operation counts ignored the cost of addition, subtraction and the cost of multiplying a finite field element by a small integer. Therefore, it would be desirable to implement the algorithms to verify their relative efficiency.

[2]Families of elliptic curves with embedding degree 18 and 24 are given in [7].

TABLE 5. A comparison of exponentiation costs in $G_{\Phi_6(q)} \subset \mathbb{F}_{q^6}^*$. $\exp_{12345}$ and $\exp_{\mathrm{GS}}$ are the exponentiation costs per bit of exponent, where the exponentiation is performed based on SQR_{12345} and GS-squaring, respectively. The exponent is represented in width-w NAF and $q = p^i \equiv 1 \pmod 6$.

(i, w)	$(2,2)$	$(3,2)$	$(4,2)$	$(2,3)$	$(3,3)$	$(4,3)$
$\exp_{12345}$	35.6	71.6	107	30.5	60.75	91.5
$\exp_{\mathrm{GS}}$	36	69	108	31.5	60	94.5

Acknowledgment

The author would like to thank Diego F. Aranha and Alfred Menezes for fruitful discussion and for their useful comments on the paper.

References

[1] D. Aranha, K. Karabina, P. Longa, C. Gebotys, and J. López. Faster explicit formulas for computing pairings over ordinary curves. 2010. Available at `http://eprint.iacr.org/2010/526`.

[2] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. *Selected Areas in Cryptography – SAC 2005, Lecture Notes in Computer Science*, 3897:319–331, 2006.

[3] N. Benger and M. Scott. Constructing tower extensions of finite fields for implementation of pairing-based cryptography. *Arithmetic of Finite Fields – WAIFI 2010, Lecture Notes in Computer Science*, 6087:180–195, 2010.

[4] J. Beuchat, J. Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya. High-speed software implementation of the optimal Ate pairing over Barreto-Naehrig curves. 2010. Available at `http://eprint.iacr.org/2010/354`.

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

[6] J. Chung and M. Hasan. Asymmetric squaring formulae. *18th IEEE Symposium on Computer Arithmetic – ARITH 2007*, pages 113–122, 2007.

[7] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23:224–280, 2010.

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

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

[10] R. Granger, D. Page, and N. Smart. High security pairing-based cryptography revisited. *Curves over Finite Fields and Applications – Algorithmic Number Theory, Lecture Notes in Computer Science*, 4076:480–494, 2006.

[11] R. Granger, D. Page, and M. Stam. A comparison of CEILIDH and XTR. *Algorithmic Number Theory symposium – ANTS VI, Lecture Notes in Computer Science*, 3076:235–249, 2004.

[12] R. Granger and M. Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. *Public Key Cryptography – PKC 2010, Lecture Notes in Computer Science*, 6056:209–223, 2010.

[13] D. Harris. Simultaneous field divisions: an extension of Montgomery's trick. 2008. Available at `http://eprint.iacr.org/2008/199`.

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

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

[16] K. Karabina. Torus-based compression by factor 4 and 6. 2010. Available at `http://eprint.iacr.org/2010/525`.

[17] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. *Tenth IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science*, 3796:13–36, 2005.

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

[19] C. Lim and H. Hwang. Fast implementation of elliptic curve arithmetic in $GF(p^n)$. *Public Key Cryptography – PKC 2000, Lecture Notes in Computer Science*, 1751:405–421, 2000.

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

[21] Y. Nogami, M. Akane, Yumi Sakemi, H. Kato, and Y. Morikawa. Integer variable χ-based ate pairing. *Pairing-Based Cryptography – Pairing 2008, Lecture Notes in Computer Science*, 5209:178–191, 2008.

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

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

[24] M. Scott. Implementing cryptographic pairings. *Pairing-Based Cryptography – Pairing 2007, Lecture Notes in Computer Science*, 4575:177–196, 2007.

[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:2843–2850, 2008.

[26] P. Smith and C. Skinner. A public-key cryptosystem and a digital signature system based on the Lucas function analogue to discrete logarithms. *Advances in Cryptology – ASIACRYPT '94, Lecture Notes In Computer Science*, 917:357–364, 1994.

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

[28] M. Stam and A. Lenstra. Efficient subgroup exponentiation in quadratic and sixth degree extensions. *Cryptographic Hardware and Embedded Systems – CHES 2002*, 2523:159–174, 2003.

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

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

Appendix A

A.1. $\mathcal{H}_0 = \{S_{0,j} : \ j = 1, 2, \ldots, 13\}$.

$$
\begin{aligned}
S_{0,1} &= 2g_0^2 + 4g_3g_4c - 1 \\
S_{0,2} &= -6g_0^2 + 4g_0 + 12g_2g_5c + 3 \\
S_{0,3} &= 3g_0^2 - 2g_0 + 3g_1^2c \\
S_{0,4} &= 1 + (2g_1^2 + 4g_2g_5)c \\
S_{0,5} &= (3g_2g_5 + 3g_3g_4)c + g_0 \\
S_{0,6} &= (-6g_1^2 + 12g_3g_4)c - 3 + 4g_0 \\
S_{0,7} &= 27g_1^2g_3^3c^3 + ((-27g_1g_2g_3^2 - 72g_1^4)g_0 + 81g_1^2g_2^2g_3 - 48g_1^4 \\
&\quad +27g_1g_2g_3^2 + 18g_3^3)c^2 + ((-9g_1g_2^3 - 42g_1^2)g_0 - 36g_1^2 \\
&\quad +6g_2^2g_3 + 21g_1g_2^3)c - 3 + 4g_0 \\
S_{0,8} &= -3/2g_1g_5^3c^3 + (-9/2g_0g_4g_5^2 - 12g_1^4 + 21/2g_4g_5^2 + 63/2g_1g_4^2g_5)c^2 \\
&\quad +((12g_1^2 - 27/2g_4^3)g_0 + 27/2g_4^3 - 9g_1^2)c - 3 + 4g_0 \\
S_{0,9} &= (63/2g_1g_2g_3^2 - 12g_1^4 + 27/2g_3^3 - 27/2g_0g_3^3)c^2
\end{aligned}
$$

$$
\begin{aligned}
&+((-9/2g_2^2g_3+12g_1^2)g_0-3/2g_1g_2^3+21/2g_2^2g_3-9g_1^2)c\\
&-3+4g_0\\
S_{0,10} &= -729/4g_1^2g_3^6c^5+(324g_1^4g_3^3+243/4g_1g_2g_3^5-243/2g_1^2g_2^2g_3^4\\
&+486g_0g_1^4g_3^3+243/16g_3^6-162g_1^5g_2g_3^2)c^4\\
&+(81g_0g_1^2g_3^3+144g_1^6+54g_1^4g_2^2g_3-513g_1^3g_2g_3^2\\
&+(-81/4g_2^4g_3^2-297/2g_3^3)g_1^2+81/2g_1g_2^3g_3^3+81/8g_2^2g_3^4)c^3\\
&+((-144g_1^4+27/8g_3^3)g_0+132g_1^4+18g_1^3g_2^3-225/2g_1^2g_2^2g_3\\
&+(27/4g_2^5g_3+207/8g_2g_3^2)g_1+27/16g_2^4g_3^2+117/8g_3^3)c^2\\
&+(6g_2^2g_3-96g_0g_1^2+12g_1g_2^3+18g_1^2)c-3+4g_0\\
S_{0,11} &= (27/4g_1g_4g_5^5-81/4g_1^2g_4^2g_5^4)c^5\\
&+(-162g_1^5g_4^2g_5+54g_1^4g_4g_5^2-243/2g_1^2g_4^4g_5^2\\
&+27/16g_4^2g_5^4+81/2g_1g_4^3g_5^3+18g_1^3g_5^3)c^4\\
&+(486g_0g_1^4g_4^3+144g_1^6+324g_1^4g_4^3-513g_1^3g_4^2g_5\\
&+(-225/2g_4g_5^2-729/4g_4^6)g_1^2+(243/4g_4^5g_5+12g_5^3)g_1\\
&+81/8g_4^4g_5^2)c^3+((-144g_1^4+81g_1^2g_4^3)g_0+207/8g_1g_4^2g_5\\
&+243/16g_4^6-297/2g_1^2g_4^3+6g_4g_5^2+132g_1^4)c^2\\
&+((-96g_1^2+27/8g_4^3)g_0+117/8g_4^3+18g_1^2)c-3+4g_0\\
S_{0,12} &= (-729/2g_1^3g_2g_3^5+972g_1^6g_3^3)c^5\\
&+(1620g_1^5g_2g_3^2-243g_1^3g_2^3g_3^3+2592g_1^6g_2^2g_3-2592g_0g_1^8\\
&+243/8g_1g_2g_3^5+243/2g_1^2g_2^2g_3^4+486g_1^4g_3^3-1728g_1^8)c^4\\
&+((-1080g_1^6-324g_1^5g_2^3)g_0-1080g_1^6+864g_1^5g_2^3-540g_1^4g_2^2g_3\\
&+(-81/2g_2^5g_3-243g_2g_3^2)g_1^3\\
&+(-621/4g_3^3+81g_2^4g_3^2)g_1^2+81/4g_1g_2^3g_3^3)c^3\\
&+((-54g_1^3g_2^3+522g_1^4)g_0+168g_1^4-135g_1^3g_2^3\\
&+(27/2g_2^6-54g_2^2g_3)g_1^2+(27g_2g_3^2+27/8g_2^5g_3)g_1+18g_3^3)c^2\\
&+((-165/2g_1^2-9/4g_1g_2^3)g_0+57/4g_1g_2^3+6g_2^2g_3+9/2g_1^2)c\\
&-3+4g_0\\
S_{0,13} &= (-81/2g_1^3g_4g_5^5+27/2g_1^2g_5^6)c^6+(81g_1^2g_4^2g_5^4+27/8g_1g_4g_5^5\\
&-324g_0g_1^5g_5^3-243g_1^3g_4^3g_5^3+2592g_1^6g_4g_5^2+864g_1^5g_5^3)c^5\\
&+((-54g_1^3g_5^3-2592g_1^8)g_0-1728g_1^8+972g_1^6g_4^3+1620g_1^5g_4^2g_5\\
&-540g_1^4g_4g_5^2+(-135g_5^3-729/2g_4^5g_5)g_1^3+243/2g_1^2g_4^4g_5^2\\
&+81/4g_1g_4^3g_5^3)c^4+((-9/4g_1g_5^3-1080g_1^6)g_0-1080g_1^6+486g_1^4g_4^3\\
&-243g_1^3g_4^2g_5-54g_1^2g_4g_5^2+(243/8g_4^5g_5+57/4g_5^3)g_1)c^3\\
&+(27g_1g_4^2g_5+6g_4g_5^2+168g_1^4+522g_0g_1^4-621/4g_1^2g_4^3)c^2\\
&+(-165/2g_0g_1^2+18g_4^3+9/2g_1^2)c-3+4g_0
\end{aligned}
$$

A.2. $\mathcal{H}_1=\{S_{1,j}:\ j=1,2,\ldots,14\}$.

$$
\begin{aligned}
S_{1,1} &= 6g_0g_1+2g_1\\
S_{1,2} &= -g_1+3g_2g_4+3g_3g_5c
\end{aligned}
$$

$$
\begin{aligned}
S_{1,3} &= (12g_1g_2g_5 + 12g_1^3 - 36g_4^2g_5)c - 24g_2g_4 + 8g_1 + 12g_2^3 \\
S_{1,4} &= (-36g_1g_3g_4 + 12g_1^3 + 36g_4^2g_5)c + 24g_2g_4 + 8g_1 - 12g_2^3 \\
S_{1,5} &= 12g_5^3c^2 + (12g_1^3 + 12g_1g_2g_5 - 36g_2g_3^2 - 24g_3g_5)c + 8g_1 \\
S_{1,6} &= 3/2g_5^3c^2 + (-6g_1g_2g_5 + 9/2g_4^2g_5)c - g_1 + 3g_2g_4 \\
S_{1,7} &= 4/3g_5^3c^2 + (4/3g_1^3 - 4g_1g_2g_5)c + 4/3g_2^3 \\
S_{1,8} &= (-6g_1g_2g_5 + 9/2g_2g_3^2 + 3g_3g_5)c - g_1 + 3/2g_2^3 \\
S_{1,9} &= (12g_1^3 + 9/2g_2g_3^2 - 18g_1g_3g_4)c - 3g_2g_4 + 8g_1 + 3/2g_2^3 \\
S_{1,10} &= 12g_5^3c^2 + (-18g_0g_3g_5 + 9/2g_2g_3^2 - 3g_3g_5)c - g_1 + 3/2g_2^3 \\
S_{1,11} &= 3/2g_5^3c^2 + (12g_1^3 - 18g_1g_3g_4 - 3g_3g_5 + 9/2g_4^2g_5)c + 8g_1 \\
S_{1,12} &= -18g_0g_2g_4 - g_1 + 12g_2^3 - 3g_2g_4 + 9/2g_4^2g_5c + 3/2g_5^3c^2 \\
S_{1,13} &= (-9/2g_1g_4g_5^2 + 3/4g_5^3)c^2 \\
&\quad +(-27/2g_1g_4^3 + 6g_1^3 + 18g_1^2g_2g_4 + 9/4g_4^2g_5)c \\
&\quad +7/2g_1 + 3/2g_2g_4 \\
S_{1,14} &= (18g_1^2g_3g_5 - 27/2g_1g_3^3)c^2 \\
&\quad +(-9/2g_1g_2^2g_3 + 3/2g_3g_5 + 6g_1^3 + 9/4g_2g_3^2)c \\
&\quad +7/2g_1 + 3/4g_2^3
\end{aligned}
$$

A.3. $\mathcal{H}_2 = \{S_{2,j} : \ j = 1, 2, \ldots, 7\}$.

$$
\begin{aligned}
S_{2,1} &= 2g_2 + 6g_4g_5c \\
S_{2,2} &= 3g_0g_2 + 3g_1g_3c - g_2 \\
S_{2,3} &= (-12g_1g_5^2 + 9g_3^2g_5)c^2 + 3g_2^2g_5c + 2g_2 \\
S_{2,4} &= (3g_1g_3 + 3g_2^2g_5 - 9g_2g_3g_4 + 6g_1^2g_2)c + 2g_2 \\
S_{2,5} &= (18g_1g_2g_3g_5 - 27/2g_2g_3^3)c^2 \\
&\quad +(6g_1^2g_2 - 9/2g_2^3g_3 + 3g_2^2g_5 + 3g_1g_3)c + 2g_2 \\
S_{2,6} &= (81/2g_1g_3^4 - 54g_1^2g_3^2g_5)c^3 \\
&\quad +(-18g_1^3g_3 - 27/4g_2g_3^3 + 9/2g_3^2g_5 + 27/2g_1g_2^2g_3^2)c^2 \\
&\quad +(-9/4g_2^3g_3 + 3g_2^2g_5 + 6g_1^2g_2 - 21/2g_1g_3)c + 2g_2 \\
S_{2,7} &= 2187/8g_3^8g_5c^6 \\
&\quad +(243g_2^2g_3^6g_5 - 2187/2g_1^2g_2g_3^6 - 8019/8g_1g_3^7 + 486g_1^5g_3^4)c^5 \\
&\quad +(-4293/4g_1g_2^2g_3^5 - 81g_1^4g_2g_3^3 + 1215/8g_2g_3^6 - 243/2g_1^2g_2^3g_3^4 \\
&\quad +567/8g_2^4g_3^4g_5 + 108g_1^5g_2^2g_3^2 + 1539/2g_1^3g_3^4 - 891/4g_3^5g_5)c^4 \\
&\quad +(-216g_1^5g_3 - 18g_1^4g_2^3g_3 + 198g_1^3g_2^2g_3^2 \\
&\quad +(27g_2^5g_3^2 + 1377/4g_2g_3^3)g_1^2 + (2997/4g_3^4 - 1593/8g_2^4g_3^3)g_1 \\
&\quad +27/4g_2^6g_3^2g_5 - 603/4g_2^2g_3^3g_5 + 54g_2^3g_3^4)c^3 \\
&\quad +((6g_2^4 - 342g_3)g_1^3 - 189/2g_1^2g_2^3g_3 \\
&\quad +(6g_2^5g_5 + 2241/4g_2^2g_3^2 - 9/4g_2^6g_3)g_1 \\
&\quad -57/2g_2^4g_3g_5 + 45g_3^2g_5 - 54g_2g_3^3 + 27/8g_2^5g_3^2)c^2 \\
&\quad +(6g_1^2g_2 + (-132g_3 + 9/2g_2^4)g_1 + 18g_2^3g_3 - 3/4g_2^7 + 3g_2^2g_5)c + 2g_2
\end{aligned}
$$

A.4. $\mathcal{H}_3 = \{S_{3,j} : \ j = 1, 2, \ldots, 10\}$.

$$
\begin{aligned}
S_{3,1} &= 4g_1g_2 + 2g_5^2c \\
S_{3,2} &= 4g_0g_3 + 2g_4^2 \\
S_{3,3} &= 3g_0g_3 + 3g_1g_2 + g_3 \\
S_{3,4} &= -2g_3 + 3g_4^2 + 3g_5^2c \\
S_{3,5} &= 12g_0g_3 + 4g_3 - 6g_5^2c \\
S_{3,6} &= 12g_1g_2 + 4g_3 - 6g_4^2 \\
S_{3,7} &= (-27/2g_3^4 + 18g_1g_3^2g_5)c^2 + (6g_1^2g_3 - 9/2g_2^2g_3^2 + 3g_2g_3g_5)c \\
&\quad +3g_1g_2 + 4g_3 \\
S_{3,8} &= (6g_1^2g_3 + 9/2g_2^2g_3^2 - 18g_1g_2g_3g_4 - 9g_3^2g_4 + 12g_1^3g_2)c \\
&\quad +3/2g_2^4 - 6g_2^2g_4 + 12g_1g_2 + 4g_3 \\
S_{3,9} &= (54g_1^2g_3^2g_4 - 27/2g_1g_2g_3^3 + 27/4g_3^4 - 36g_1^4g_3)c^2 \\
&\quad +(-9/2g_1g_2^3g_3 + 12g_1^3g_2 - 21g_1^2g_3 + 27/4g_2^2g_3^2 - 27/2g_3^2g_4)c \\
&\quad +3/2g_2^4 - 6g_2^2g_4 + 12g_1g_2 + 4g_3 \\
S_{3,10} &= (729/4g_1^2g_3^7 - 2187/8g_3^8g_4)c^5 \\
&\quad +(-243g_2^2g_3^6g_4 + 81g_1^4g_3^4 - 243/8g_3^7 + 1053/2g_1^2g_2^2g_3^5 \\
&\quad +243/2g_1g_2g_3^6 - 162g_1^5g_2g_3^3)c^4 \\
&\quad +(-36g_1^5g_2^3g_3 + 72g_1^4g_2^2g_3^2 - 567/2g_1^3g_2g_3^3 \\
&\quad +(351/4g_2^4g_3^3 - 81/4g_3^4)g_1^2 + 999/4g_1g_2^3g_3^4 \\
&\quad -567/8g_2^4g_3^4g_4 - 513/4g_2^2g_3^5 + 891/4g_3^5g_4)c^3 \\
&\quad +((12g_2^4 - 36g_3)g_1^4 - 81g_1^3g_2^3g_3 + (-9/2g_2^6g_3 + 27g_2^2g_3^2)g_1^2 \\
&\quad +(-27g_2g_3^3 + 225/4g_2^5g_3^2)g_1 - 459/8g_2^4g_3^3 \\
&\quad +603/4g_2^2g_3^3g_4 - 27/4g_2^6g_3^2g_4 + 27/2g_3^4)c^2 \\
&\quad +(12g_1^3g_2 + (-21g_3 + 9g_2^4)g_1^2 + (-45g_2^3g_3 + 3/2g_2^7 - 6g_2^5g_4)g_1 \\
&\quad +27g_2^2g_3^2 - 54g_3^2g_4 + 57/2g_2^4g_3g_4 - 15/2g_2^6g_3)c \\
&\quad +3/2g_2^4 - 6g_2^2g_4 + 12g_1g_2 + 4g_3
\end{aligned}
$$

A.5. $\mathcal{H}_4 = \{S_{4,j} : \ j = 1, 2, \ldots, 10\}$.

$$
\begin{aligned}
S_{4,1} &= 4g_0g_4 + 2g_3^2c \\
S_{4,2} &= 4g_1g_5c + 2g_2^2 \\
S_{4,3} &= 3g_0g_4 + 3g_1g_5c + g_4 \\
S_{4,4} &= 12g_0g_4 - 6g_2^2 + 4g_4 \\
S_{4,5} &= 3g_2^2 + 3g_3^2c - 2g_4 \\
S_{4,6} &= (12g_1g_5 - 6g_3^2)c + 4g_4 \\
S_{4,7} &= -9/2g_4^2g_5^2c^2 + (6g_1^2g_4 + (3g_5 + 18g_2g_4^2)g_1 \\
&\quad +3g_2g_4g_5 - 27/2g_4^4)c + 4g_4 \\
S_{4,8} &= 3/2g_5^4c^3 + (-18g_1g_3g_4g_5 + 12g_1^3g_5 - 6g_3g_5^2 + 9/2g_4^2g_5^2)c^2 \\
&\quad +(-9g_3g_4^2 + 6g_1^2g_4 + 12g_1g_5)c + 4g_4
\end{aligned}
$$

$$
\begin{aligned}
S_{4,9} &= (-9/2g_1g_4g_5^3 + 3/2g_5^4)c^3 \\
&\quad +(-36g_1^4g_4 - 27/2g_1g_4^3g_5 + 12g_1^3g_5 \\
&\quad +54g_1^2g_3g_4^2 - 6g_3g_5^2 + 27/4g_4^2g_5^2)c^2 \\
&\quad +(27/4g_4^4 + 12g_1g_5 - 21g_1^2g_4 - 27/2g_3g_4^2)c + 4g_4 \\
S_{4,10} &= (-9/2g_1^2g_4g_5^6 - 27/4g_3g_4^2g_5^6 + 3/2g_1g_5^7)c^6 \\
&\quad +(-36g_1^5g_4g_5^3 + 12g_1^4g_5^4 + 351/4g_1^2g_4^3g_5^4 \\
&\quad +(-6g_3g_5^5 + 225/4g_4^2g_5^5)g_1 - 567/8g_3g_4^4g_5^4 - 15/2g_4g_5^6)c^5 \\
&\quad +(-162g_1^5g_4^3g_5 + 72g_1^4g_4^2g_5^2 - 81g_1^3g_4g_5^3 \\
&\quad +(9g_5^4 + 1053/2g_4^5g_5^2)g_1^2 + 999/4g_1g_4^4g_5^3 \\
&\quad +(57/2g_4g_5^4 - 243g_4^6g_5^2)g_3 - 459/8g_4^3g_5^4)c^4 \\
&\quad +(81g_1^4g_4^4 - 567/2g_1^3g_4^3g_5 + (729/4g_4^7 + 27g_4^2g_5^2)g_1^2 \\
&\quad +(-45g_4g_5^3 + 243/2g_4^6g_5)g_1 + (-2187/8g_4^8 + 603/4g_4^3g_5^2)g_3 \\
&\quad +3/2g_5^4 - 513/4g_4^5g_5^2)c^3 + (-36g_1^4g_4 + 12g_1^3g_5 \\
&\quad -81/4g_1^2g_4^4 - 27g_1g_4^3g_5 + (-6g_5^2 + 891/4g_4^5)g_3 \\
&\quad +27g_4^2g_5^2 - 243/8g_4^7)c^2 + (12g_1g_5 + 27/2g_4^4 - 21g_1^2g_4 - 54g_3g_4^2)c \\
&\quad +4g_4
\end{aligned}
$$

A.6. $\mathcal{H}_5 = \{S_{5,j} : \ j = 1, 2, \ldots, 7\}$.

$$
\begin{aligned}
S_{5,1} &= 6g_2g_3 + 2g_5 \\
S_{5,2} &= 3g_0g_5 + 3g_1g_4 - g_5 \\
S_{5,3} &= -12g_1g_2^2 + 9g_2g_4^2 + 3g_2g_5^2c + 2g_5 \\
S_{5,4} &= (3g_2g_5^2 - 9g_3g_4g_5 + 6g_1^2g_5)c + 2g_5 + 3g_1g_4 \\
S_{5,5} &= -9/2g_4g_5^3c^2 + (6g_1^2g_5 + 18g_1g_2g_4g_5 + 3g_2g_5^2 - 27/2g_4^3g_5)c \\
&\quad +2g_5 + 3g_1g_4 \\
S_{5,6} &= (27/2g_1g_4^2g_5^2 - 9/4g_4g_5^3)c^2 \\
&\quad +(-18g_1^3g_4 + (-54g_2g_4^2 + 6g_5)g_1^2 + 81/2g_1g_4^4 + 3g_2g_5^2 - 27/4g_4^3g_5)c \\
&\quad -21/2g_1g_4 + 9/2g_2g_4^2 + 2g_5 \\
S_{5,7} &= (27/4g_2g_4^2g_5^6 + 27g_1^2g_4^2g_5^5 - 3/4g_5^7 - 9/4g_1g_4g_5^6)c^5 \\
&\quad +(108g_1^5g_4^2g_5^2 - 18g_1^4g_4g_5^3 + 6g_1^3g_5^4 - 243/2g_1^2g_4^4g_5^3 \\
&\quad +(-1593/8g_4^3g_5^4 + 6g_2g_5^5)g_1 + 27/8g_4^2g_5^5 + 567/8g_2g_4^4g_5^4)c^4 \\
&\quad +(486g_1^5g_4^4 - 81g_1^4g_4^3g_5 + 198g_1^3g_4^2g_5^2 \\
&\quad +(-2187/2g_4^6g_5 - 189/2g_4g_5^3)g_1^2 + (9/2g_5^4 - 4293/4g_4^5g_5^2)g_1 \\
&\quad +(-57/2g_4g_5^4 + 243g_4^6g_5^2)g_2 + 54g_4^4g_5^3)c^3 \\
&\quad +(-216g_1^5g_4 + 1539/2g_1^3g_4^4 + 1377/4g_1^2g_4^3g_5 \\
&\quad +(2241/4g_4^2g_5^2 - 8019/8g_4^7)g_1 + (-603/4g_4^3g_5^2 + 2187/8g_4^8)g_2 \\
&\quad +18g_4g_5^3 + 1215/8g_4^6g_5)c^2 + (-342g_1^3g_4 + 6g_1^2g_5 + 2997/4g_1g_4^4 \\
&\quad +(3g_5^2 - 891/4g_4^5)g_2 - 54g_4^3g_5)c + 2g_5 - 132g_1g_4 + 45g_2g_4^2
\end{aligned}
$$

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

E-mail address: kkarabin@uwaterloo.ca