Some Efficient Algorithms for the Final Exponentiation of η_T Pairing

Masaaki Shirase[1], Tsuyoshi Takagi[1], and Eiji Okamoto[2]

[1] Future University-Hakodate, Japan
[2] University of Tsukuba, Japan

Abstract. Recently Tate pairing and its variations are attracted in cryptography. Their operations consist of a main iteration loop and a final exponentiation. The final exponentiation is necessary for generating a unique value of the bilinear pairing in the extension fields. The speed of the main loop has become fast by the recent improvements, *e.g.*, the Duursma-Lee algorithm and η_T pairing. In this paper we discuss how to enhance the speed of the final exponentiation of the η_T pairing in the extension field $\mathbb{F}_{3^{6n}}$. Indeed, we propose some efficient algorithms using the torus $T_2(\mathbb{F}_{3^{3n}})$ that can efficiently compute an inversion and a powering by 3^n+1. Consequently, the total processing cost of computing the η_T pairing can be reduced by 17% for $n = 97$.

Keywords: Tate pairing, η_T pairing, final exponentiation, torus

1 Introduction

Bilinear pairings deliver us new cryptographic applications such as identity-based encryptions [3], short signatures [5], and efficient broadcast encryptions [4]. Recently Duursma and Lee [6] proposed an efficient algorithm for computing Tate pairing. The Duursma-Lee algorithm uses the supersingular curves,

$$E^b(\mathbb{F}_{3^n}) : y^2 = x^3 - x + b \text{ with } b \in \{-1, 1\}. \tag{1}$$

Kwon proposed an efficient variation of the Duursma-Lee algorithm that requires no cube root operation [10]. Barreto *et. al.* proposed the η_T pairing [1], which reduces the number of the main loop in the Duursma-Lee algorithm to half. Beuchat *et. al.* presented a faster variation of η_T pairing without a cube root operation [2]. Currently the η_T pairing is one of the fastest algorithms for computing the bilinear pairing.

Both the Duursma-Lee algorithm and the η_T pairing require the "final exponentiation", *i.e.*, A^s for $A \in \mathbb{F}_{3^{6n}}$ and some integer s, since the resulting element by the pairing algorithms is contained in the quotient group $\mathbb{F}_{3^{6n}}^*/\mathbb{F}_{3^{3n}}^*$. The final exponentiations for the Duursma-Lee algorithm and the η_T pairing are $A^{3^{3n}-1}$ and A^W with $W = (3^{3n}-1)(3^n+1)(3^n+1-b3^{(n+1)/2})$, respectively. The η_T pairing without the final exponentiation is about twice faster than the Duursma-Lee algorithm, but the final exponentiation in the η_T pairing causes a relatively large overhead. For example, Shu *et. al.* [12] estimated that the η_T

pairing with the final exponentiation is as fast as the Duursma-Lee algorithm in hardware. Ronan *et. al.* reported that the straightforward implementation of the final exponentiation is more than 35% of the whole algorithm [11]. In Section 4, we estimated that the currently fastest final exponentiation [2] is about 26% of the whole algorithm.

In this paper we try to reduce the cost of the final exponentiation of the η_T pairing. Note that $A^{3^{3n}-1}$ is an element in the torus $T_2(\mathbb{F}_{3^{3n}})$, which is a subgroup of $\mathbb{F}^*_{3^{6n}}$. We show that an inversion and an powering by (3^n+1)-th in $T_2(\mathbb{F}_{3^{3n}})$ are efficiently computed for the basis $\{1, \sigma\}$ of $\mathbb{F}_{3^{6n}}$ over $\mathbb{F}_{3^{3n}}$ with $\sigma^2 + 1 = 0$. We then present an efficient algorithm for the final exponentiation $A^W = B^{(3^n+1)(3^n+1-b3^{(n+1)/2})}$ with $B = A^{3^{3n}-1}$ of the η_T pairing in the torus $T_2(\mathbb{F}_{3^{3n}})$, which can be computed with 36 multiplications in $\mathbb{F}_{3^n}$ plus other negligible operations. Consequently, the final exponentiation of our proposed scheme requires only about 13% of the whole η_T pairing, which achieves about 17% faster η_T pairing than the previous known algorithms.

On the other hand, Granger *et. al.* presented an encoding method of $\mathbb{F}^*_{3^{6n}}/\mathbb{F}^*_{3^{3n}}$ [8], which eliminates the final exponentiation from the Duursma-Lee algorithm. We call it the GPS encoding according to the authors' name. In this paper, we discuss how to apply the GPS encoding to the η_T pairing. The η_T pairing with the GPS encoding can be faster depending on the information of b.

The remainder of this paper is organized as follows: In Section 2 we explain Tate pairing and the η_T pairing. In Section 3 we describe several representations (including the Torus $T_2(\mathbb{F}_{3^{3n}})$ and the GPS encoding) of group $\mathbb{F}^*_{3^{6n}}$ and apply them to the efficient computation of the final exponentiation. In Section 4 we propose new efficient algorithms of computing the final exponentiation for the η_T pairing and how to apply the GPS encoding to the η_T pairing. In Section 5 we conclude this paper.

2 Tate Pairing and η_T Pairing

In this section we explain about Tate pairing and its efficient variations, namely the Duursma-Lee algorithm and the η_T pairing.

2.1 Tate Pairing

Let $\mathbb{F}_q$ be a finite field with q elements, where q be a power of the characteristic p. Let E be elliptic curves defined over $\mathbb{F}_q$, and let $\mathcal{O}_E$ be the point at infinity. Let l be a positive integer relatively prime to q with $l \,|\, \#E(\mathbb{F}_q)$, and let k be the minimal positive integer with $l \,|\, (q^k - 1)$. This k is called the embedded degree. Then Tate pairing is a map

$$\langle \cdot, \cdot \rangle_l : E(\mathbb{F}_q)[l] \times E(\mathbb{F}_{q^k})/lE(\mathbb{F}_{q^k}) \to \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^l,$$

which satisfies the bilinearity $\langle P, aQ \rangle_l = \langle aP, Q \rangle_l = \langle P, Q \rangle_l^a$ for any integer $a \neq 0$, and is non-degenerate, *i.e.*, there exists a $Q \in E(\mathbb{F}_{q^k})$ such that $\langle P, Q \rangle_l \notin (\mathbb{F}^*_{q^k})^l$ for $P \in E(\mathbb{F}_{q^k})[l] \setminus \{\mathcal{O}_E\}$.

It is typically selected that l and q^k are about 160 bits and 1024 bits, respectively. One of the most efficient classes for computing the bilinear map is constructed over supersingular elliptic curves. The embedded degree k of supersingular elliptic curves is one of 4, 6, or 2 for characteristic 2, 3 or $p > 3$, respectively. This paper deals with the case of characteristic 3 and uses elliptic curves formed by (1). It is known that $\#E^b(\mathbb{F}_{3^n}) = 3^n + 1 + b'3^{(n+1)/2}$, where b' is defined as

$$b' = \begin{cases} b & \text{if } n \equiv 1 \bmod 12, \\ -b & \text{if } n \equiv 7 \bmod 12. \end{cases}$$

Note that we have $n \equiv \pm 1,\ \pm 5 \bmod 12$ since n has to be coprime to 6 [6].

We require an injection ψ from $E(\mathbb{F}_{3^n})[l]$ to $E(\mathbb{F}_{3^{6n}})/lE(\mathbb{F}_{3^{6n}})$ since $\langle P, Q\rangle_l$ is defined for points $P \in E(\mathbb{F}_{3^n})[l]$ and $Q \in E(\mathbb{F}_{3^{6n}})/lE(\mathbb{F}_{3^{6n}})$. This ψ is sometimes called as the distortion map. In the case of characteristic three, the distortion map is defined as $\psi(x, y) = (-x + \rho, y\ \sigma)$ for $(x, y) \in E(\mathbb{F}_{3^n})$, where σ and ρ satisfy

$$\sigma^2 = -1 \text{ and } \rho^3 = \rho + b'.$$

We usually select the basis $\{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ of $\mathbb{F}_{3^{6n}}$ over $\mathbb{F}_{3^n}$, where ρ and σ are utilized in the distortion map. Every element A in $\mathbb{F}_{3^{6n}}$ is then represented as $A = a_0 + a_1\sigma + a_2\rho + a_3\sigma\rho + a_4\rho^2 + a_5\sigma\rho^2$ for some $a_i \in \mathbb{F}_{3^n}$. Moreover an element A_0 in $\mathbb{F}_{3^{3n}}$ is represented as $A_0 = a_0 + a_2\rho + a_4\rho^2$. We denote by M_k, C_k and I_k the computational cost of multiplication, cubing, and inversion in $\mathbb{F}_{3^{kn}}$, respectively. Then the following relationships

$$M_6 = 3M_3,\ M_3 = 6M_1,\ C_6 = 2C_3,\ C_3 = 3C_1,\ I_6 = 5M_3 + I_3,\ I_3 = 8M_1 + I_1 \tag{2}$$

are held [9]. The computational costs appeared in this paper are estimated using Eq. (2). The computational cost is estimated without considering the costs of addition and subtraction which are usually negligible. Beuchat *et. al.* pointed out A^{3^n} in $\mathbb{F}_{3^{6n}}$ can be computed virtually for free [2] (see Appendix B). Therefore we have

$$\text{the cost of } A^{3^n+1} (= A^{3^n} \cdot A) \text{ is } M_6 = 18M_1. \tag{3}$$

The resulting value $\langle P, \psi(Q)\rangle_l$ of Tate pairing is contained in the quotient group $\mathbb{F}^*_{3^{6n}}/(\mathbb{F}^*_{3^{6n}})^l$. Then there are many choices for representing elements in a coset of the quotient group. Indeed $A, B \in \mathbb{F}^*_{3^{6n}}$ are contained in the same coset, if they satisfies $B = A \cdot C^l$ for some $C \in \mathbb{F}^*_{3^{6n}}$. We are able to eliminate this *ambiguity* by using the final exponentiation. The final exponentiation tries to compute the $((3^{6n} - 1)/l)$-th powering to the output from the Tate pairing. Therefore we also deploy the modified Tate pairing $\hat{e}(P, Q)$ defined by

$$\hat{e} : E(\mathbb{F}_{3^n})[l] \times E(\mathbb{F}_{3^n})[l] \to \mathbb{F}^*_{3^{6n}},\ (P, Q) \mapsto \hat{e}(P, Q) = \langle P, \psi(Q)\rangle_l^{(3^{6n}-1)/l},$$

whose value in $\mathbb{F}^*_{3^{6n}}$ can be uniquely determined.

Granger *et. al.* proposed another technique to remove the ambiguity [8]. In this paper we denote by *GPS encoding* the technique proposed by Granger *et. al.* according to the authors' name (refer Sections 3.2).

Algorithm 1: Duursma-Lee Algorithm [10]

input: $P = (x_p, y_p),\ Q = (x_q, y_q) \in E^b(\mathbb{F}_{3^n})[l]$

output: $\langle P, \psi(Q)\rangle_{3^{3n}+1} \in \mathbb{F}^*_{3^{6n}}/(\mathbb{F}^*_{3^{6n}})^{3^{3n}+1}$

1: $R_0 \leftarrow 1$ (in $\mathbb{F}_{3^{6n}}$), $x_q \leftarrow x_q^3,\ y_q \leftarrow y_q^3$ (in $\mathbb{F}_{3^n}$), $d \leftarrow (bn \bmod 3)$
2: **for** $i \leftarrow 0$ **to** $n-1$ **do**
3: $\quad x_p \leftarrow x_p^9,\ y_p \leftarrow y_p^9$ (in $\mathbb{F}_{3^n}$)
4: $\quad r_0 \leftarrow x_p + x_q + d$ (in $\mathbb{F}_{3^n}$)
5: $\quad R_1 \leftarrow -r_0^2 - y_p y_q \sigma - r_0\rho - \rho^2$ (in $\mathbb{F}_{3^{6n}}$)
6: $\quad R_0 \leftarrow R_0^3$ (in $\mathbb{F}_{3^{6n}}$)
7: $\quad R_0 \leftarrow R_0 R_1$ (in $\mathbb{F}_{3^{6n}}$)
8: $\quad y_q \leftarrow -y_q$ (in $\mathbb{F}_{3^n}$)
9: $\quad d \leftarrow ((d-b) \underline{m}od 3)$
10: **end for**
11: **return** R_0 (Cost: $15nM_1 + (10n+2)C_1$)

Algorithm 2: Computation of $\eta_T(P,Q)^{3^{(n+1)/2}}$ for $n \equiv 1 \bmod 12$ [2]

input: $P = (x_p, y_p), Q = (x_q, y_q) \in E^b(\mathbb{F}_{3^n})[l]$

output: $\eta_T(P,Q)^{3^{(n+1)/2}} \in \mathbb{F}^*_{3^{6n}}/(\mathbb{F}^*_{3^{6n}})^{3^n+1+b'3^{(n+1)/2}}$

1: **if** $b = 1$ **then** $y_p \leftarrow -y_p$
2: $d \leftarrow b$ (in $\mathbb{F}_3$), $R_0 \leftarrow -y_p(x_p + x_q + b) + y_q\sigma + y_p\rho$ (in $\mathbb{F}_{3^{6n}}$)
3: **for** $i \leftarrow 0$ **to** $(n-1)/2$ **do**
4: $\quad r_0 \leftarrow x_p + x_q + d$ (in $\mathbb{F}_{3^n}$)
5: $\quad R_1 \leftarrow -r_0^2 + y_p y_q \sigma - r_0\rho - \rho^2$ (in $\mathbb{F}_{3^{6n}}$)
6: $\quad R_0 \leftarrow R_0 R_1$ (in $\mathbb{F}_{3^{6n}}$)
7: $\quad y_p \leftarrow -y_p$ (in $\mathbb{F}_{3^n}$)
8: $\quad x_q \leftarrow x_q^9,\ y_q \leftarrow y_q^9$ (in $\mathbb{F}_{3^n}$)
9: $\quad R_0 \leftarrow R_0^3$ (in $\mathbb{F}_{3^{6n}}$
10: $\quad d \leftarrow ((d-b) \bmod 3)$
11: **end for**
12: **return** R_0 (Cost: $(7.5n + 8.5)M_1 + (5n+5)C_1$)

2.2 Efficient Pairings on Supersingular Curves over $\mathbb{F}_{3^n}$

We explain about some efficient algorithms for computing the bilinear pairing over supersingular curves with characteristic three. Algorithm 1 is the Duursma-Lee algorithm which outputs $\langle P, \psi(Q)\rangle_{3^{3n}+1}$ [10]. The Duursma-Lee algorithm has n interactions in the main loop and the whole computational cost is $15nM_1 + (10n+2)C_1$. Note that the final exponentiation of the Duursma-Lee algorithm uses the powering to $(3^{6n}-1)/(3^{3n}+1) = (3^{3n}-1)$.

Next Barreto *et. al.* introduced the η_T pairing [1]. The η_T pairing is also defined on supersingular elliptic curves formed by (1) for $n \equiv 1 \bmod 6$ in the case of characteristic three. Beuchat *et. al.* proposed a variation of the η_T pairing (Algorithm 2), which requires no cube root calculation and outputs $\eta_T(P,Q)^{3^{(n+1)/2}}$ in the case of $n \equiv 1 \bmod 12$ [2]. The number of iterations in the main loop of

the η_T pairing becomes $(n+1)/2$, which is half for the Duursma-Lee algorithm. The computational cost of Algorithm 2 is $(7.5n+8.5)M_1 + (5n+5)C_1$.

Note that the η_T pairing itself does not satisfy the bilinearity. Therefore we have to compute the final exponentiation with W-th powering with

$$W = (3^{3n}-1)(3^n+1)(3^n+1-b'3^{(n+1)/2}) = (3^{6n}-1)/\#E^b(\mathbb{F}_{3^n}).$$

This powering function by W is the final exponentiation in the η_T pairing.

Finally we have the following relationship between the modified Tate pairing and the η_T pairing

$$(\eta_T(P,Q)^W)^{3T^2} = \hat{e}(P,Q)^Z, \tag{4}$$

where $T = -b'3^{(n+1)/2} - 1, Z = -b'3^{(n+3)/2}$ [1]. An efficient algorithm of mapping $\eta_T(P,Q)^W$ to $\hat{e}(P,Q)$ is described in Section 4.5.

3 Efficient Final Exponentiation and GPS Encoding

In this section we present the final exponentiation of Tate pairing and the GPS encoding that requires no final exponentiation.

3.1 Efficient Final Exponentiation for Duursma-Lee algorithm

We recall how to efficiently compute the final exponentiation of the Duursma-Lee algorithm, namely $A^{3^{3n}-1}$ for $A \in \mathbb{F}_{3^{6n}}$ [9].

The base of $\mathbb{F}_{3^{6n}}$ over $\mathbb{F}_{3^n}$ is fixed with $\{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ as we discussed in Section 2. Let A_0 and A_1 be elements in $\mathbb{F}_{3^{3n}}$ with $A_0 = a_0 + a_2\rho + a_4\rho^2$ and $A_1 = a_1 + a_3\rho + a_5\rho^2$. Then every element $A \in \mathbb{F}_{3^{6n}}$ is represented as

$$A = A_0 + A_1\sigma = a_0 + a_1\sigma + a_2\rho + a_3\sigma\rho + a_4\rho^2 + a_5\sigma\rho^2.$$

This means that $\mathbb{F}_{3^{6n}}$ is a quadratic extension from $\mathbb{F}_{3^{3n}}$ with the basis $\{1, \sigma\}$. It is easily to know that $\sigma^{3^{3n}} = -\sigma$ for $n \equiv 1 \bmod 6$ which is a necessary condition for the Duursma-Lee algorithm and the η_T pairing algorithms. We then have the relationship

$$A^{3^{3n}} = (A_0 + A_1\sigma)^{3^{3n}} = A_0^{3^{3n}} + A_1^{3^{3n}}\sigma^{3^{3n}} = A_0 - A_1\sigma$$

for $A = A_0 + A_1\sigma \in \mathbb{F}_{3^{6n}}^*$. Therefore, the final exponentiation for the Duursma-Lee algorithm is performed as follows:

$$A^{3^{3n}-1} = \frac{A^{3^{3n}}}{A} = \frac{A_0 - A_1\sigma}{A_0 + A_1\sigma}\ .$$

Moreover $(A_0 + A_1\sigma)\cdot(A_0 - A_1\sigma) = A_0^2 + A_1^2 \in \mathbb{F}_{3^{3n}}^*$ yields the equation

$$A^{3^{3n}-1} = \frac{(A_0 - A_1\sigma)^2}{A_0^2 + A_1^2} = \frac{(A_0^2 - A_1^2) - 2A_0A_1\sigma}{A_0^2 + A_1^2}. \tag{5}$$

Then the computational cost of the final exponentiation for the Duursma-Lee algorithm is

$$5M_3 + I_3 = 30M_1 + I_3. \tag{6}$$

Table 1. Final exponentiation and GPS encoding for the Duursma-Lee algorithm

	Output of Duursma-Lee algorithm	GPS encoding
Group	$\mathcal{G} = \mathbb{F}^*_{3^{6n}}/\mathbb{F}^*_{3^{3n}}$	$\mathbb{F}^*_{3^n} \cup \{\mathcal{O}\}$
Element	$A_0 + A_1\sigma,\ A_0, A_1 \in \mathbb{F}_{3^{3n}}$	A_0/A_1 (Cost: $6M_1 + I_3$)
Final exponentiation	$\dfrac{A_0 - A_1\sigma}{A_0 + A_1\sigma}$ (Cost: $30M_1 + I_3$)	–

GPS encoding requires no final exponentiation.

3.2 GPS Encoding in $\mathbb{F}^*_{3^{6n}}/\mathbb{F}^*_{3^{3n}}$

The GPS encoding is another technique of removing the ambiguity of representation from the cosets in a quotient group $\mathbb{F}^*_{3^{6n}}/(\mathbb{F}^*_{3^{6n}})^l$ [8].

Denote by $\mathcal{G}$ be a quotient group resulting from the Duursma-Lee algorithm, namely $\mathcal{G} = \mathbb{F}^*_{3^{6n}}/(\mathbb{F}^*_{3^{6n}})^{3^{3n}+1}$. This group $\mathcal{G}$ has a group raw which is isomorphic to a subgroup of $\mathbb{F}^*_{3^{6n}}$. We then have the relationship $\mathcal{G} = \mathbb{F}^*_{3^{6n}}/\mathbb{F}^*_{3^{3n}}$ due to $\mathbb{F}^*_{3^{3n}} = (\mathbb{F}^*_{3^{6n}})^{3^{3n}+1}$. In other words, both $A_0 + A_1\sigma$ and $(\lambda A_0) + (\lambda A_1)\sigma$ are contained in the same coset. Especially $A_0 + A_1\sigma$ is equivalent to $A_0/A_1 + \sigma$ in $\mathcal{G}$ in the case of $A_1 \neq 0$. Therefore the map

$$\tau\ :\ \mathcal{G} \to \mathbb{F}_{3^{3n}} \cup \{\mathcal{O}\},\quad A_0 + A_1\sigma \mapsto \begin{cases} A_0/A_1 & \text{if } A_1 \neq 0 \\ \mathcal{O} & \text{if } A_1 = 0 \end{cases}$$

is a bijection and gives a representation for $\mathcal{G}$ without ambiguity, where $\mathcal{O}$ is the point at infinity. This representation for $\mathcal{G}$ is called the GPS encoding in this paper. The computational cost for computing the GPS encoding for a given $A \in \mathbb{F}^*_{3^{6n}}$ is

$$M_3 + I_3 = 6M_1 + I_3,$$

because the map τ is performed by one division in $\mathbb{F}_{3^{3n}}$ (= one inversion and one multiplication).

Table 1 gives a comparison of the final exponentiation with the GPS encoding for the Duursma-Lee algorithm.

4 The Proposed Algorithm

In this section we present a new efficient final exponentiation and the GPS encoding for the η_T pairing.

4.1 Torus $T_2(\mathbb{F}_{3^{3n}})$

Granger *et. al.* introduced the torus $T_2(\mathbb{F}_{3^{3n}})$ for compressing the value of $\mathbb{F}_{3^{6n}}$ [8]. At first we describe the arithmetic of the torus.

Let L be an n-th extension field of a field k. Let $N_{L/F}$ be a norm map to field F with $k \subset F \subsetneq L$. The torus $T_m(k)$ is a subgroup of L^* defined by

$T_m(k) = \cap_{k \subset F \subsetneq L} Ker[N_{L/F}]$. In the paper we especially deal with the $T_2(k) = Ker[N_{L/k}]$ in the case of $m = 2$, $k = \mathbb{F}_{3^{3n}}$, and $L = \mathbb{F}_{3^{6n}}$. Every element in $\mathbb{F}^*_{3^{6n}}$ is represented as $A = A_0 + A_1\sigma$ with $A_0, A_1 \in \mathbb{F}_{3^{3n}}$. The conjugate element of $A = A_0 + A_1\sigma$ in $\mathbb{F}^*_{3^{6n}}$ is $\bar{A} = A_0 - A_1\sigma$, and thus $N_{\mathbb{F}_{3^{6n}}/\mathbb{F}_{3^{3n}}}(A) = A\bar{A} = A_0^2 + A_1^2$. Therefore $T_2(\mathbb{F}_{3^{3n}})$ can be represented by

$$T_2(\mathbb{F}_{3^{3n}}) = \{A_0 + A_1\sigma \in \mathbb{F}^*_{3^{6n}} \ : \ A_0^2 + A_1^2 = 1\}.$$

The element $A_0 + A_1\sigma \in \mathbb{F}_{3^{6n}}$ can be compressed to the half using the relationship $A_0^2 + A_1^2 = 1$ (Refer [8] for the further results about the compression of the pairing value).

4.2 The Proposed Final Exponentiation

We point out that some operations in the torus $T_2(\mathbb{F}_{3^{3n}})$ can be computed efficiently. We then present a new efficient final exponentiation algorithm for the η_T pairing.

At first we can easily prove the following lemma.

Lemma 1. *The torus $T_2(\mathbb{F}_{3^{3n}})$ has following properties.*
(i) $A_0 - A_1\sigma = (A_0 + A_1\sigma)^{-1}$ for $A_0 + A_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$.
(ii) $(A_0 + A_1\sigma)^{3^{3n}-1} \in T_2(\mathbb{F}_{3^{3n}})$ for $A_0 + A_1\sigma \in \mathbb{F}^*_{3^{6n}}$.

Proof. (i) $A_0 - A_1\sigma$ is the inverse of $A_0 + A_1\sigma$ due to $(A_0 + A_1\sigma)(A_0 - A_1\sigma) = A_0^2 + A_1^2 = 1$ for $A_0 + A_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$. (ii) The summation of a squaring of the constant term and that of the coefficient of Eq. (5) is equal to $\dfrac{(A_0^2 - A_1^2)^2 + (2A_0A_1)^2}{(A_0^2 + A_1^2)^2} = 1$, and thus we obtain $(A_0 + A_1)^{3^{3n}-1} \in T_2(\mathbb{F}_{3^{3n}})$. $\square$

Therefore, the computational cost of the inversion in the torus $T_2(\mathbb{F}_{3^{3n}})$ is virtually for free.

Next let $A \in \mathbb{F}_{3^{6n}}$ be an output value from the η_T pairing. Note that $B = A^{3^{3n}-1}$ is contained in the torus $T_2(\mathbb{F}_{3^{3n}})$ due to Lemma 1. Then the final exponentiation A^W with $W = (3^{3n} - 1)(3^n + 1)(3^n + 1 - b'3^{(n+1)/2})$, can be computed as follows:

$$A^W = \begin{cases} D \cdot E^{-1} & \text{if } b' = 1 \\ D \cdot E & \text{if } b' = -1, \end{cases}$$

where $D = C^{3^n+1}$ and $E = C^{3^{(n+1)/2}}$ with $C = B^{3^n+1}$. It is easily to see that C, D and $E \in T_2(\mathbb{F}_{3^{3n}})$ since $T_2(\mathbb{F}_{3^{3n}})$ is a subgroup of $\mathbb{F}^*_{3^{6n}}$. The computation of $C^{3^{(n+1)/2}}$ can be efficiently performed by repeatedly calling the cubing algorithm in $\mathbb{F}_{3^n}$. On other hand we have the following lemma for the computation of X^{3^n+1} with $X \in T_2(\mathbb{F}_{3^{3n}})$ that requires no cubing.

Algorithm 3: Proposed Final Exponentiation of η_T Pairing

input: $A = (a_0, a_1, a_2, a_3, a_4, a_5) \in \mathbb{F}^*_{3^{6n}}, \quad b' \in \{0, 1\}$
output: $A^W \in \mathbb{F}^*_{3^{6n}}$ for $W = (3^{3n} - 1)(3^n + 1)(3^n + 1 - b'3^{(n+1)/2})$

1: $B \leftarrow A^{3^{3n}-1}$ (in $\mathbb{F}_{3^{6n}}$) (Eq.(5))
2: $C \leftarrow B^{3^n+1} = \Lambda(B)$ (in $T_2(\mathbb{F}_{3^{3n}})$) (Lemma 2)
3: $D \leftarrow C^{3^n+1} = \Lambda(C)$ (in $T_2(\mathbb{F}_{3^{3n}})$) (Lemma 2)
4: $E \leftarrow C$
5: **for** $i \leftarrow 0$ **to** $(n-1)/2$ **do**
6: $\quad E \leftarrow E^3$ in $\mathbb{F}_{3^{6n}}$ (in $\mathbb{F}_{3^{6n}}$)
7: **end for**
8 **if** $(b' = 1)$ **then return** $D \cdot \overline{E}$ (in $\mathbb{F}_{3^{6n}}$) (Cost: $66M_3 + 3(n+1)C_1 + I_3$)
9: **else return** $D \cdot E$ (in $\mathbb{F}_{3^{6n}}$) (Cost: $66M_3 + 3(n+1)C_1 + I_3$)

Lemma 2. *Let $n \equiv 1 \bmod 6$. For $X = X_0 + X_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$ we can compute $Y = \Lambda(X) = X^{3^n+1} = Y_0 + Y_1\sigma$ with 9 multiplications in $\mathbb{F}_{3^n}$ as follows: Let $z_0 \sim z_8$ be defined as*

$$\begin{array}{lll} z_0 = x_0x_4, & z_1 = x_1x_5, & z_2 = x_2x_4, \\ z_3 = x_3x_5, & z_4 = (x_0 + x_1)(x_4 - x_5), & z_5 = x_1x_2, \\ z_6 = x_0x_3, & z_7 = (x_0 + x_1)(x_2 + x_3), & z_8 = (x_2 + x_3)(x_4 - x_5), \end{array}$$

then, Y can be computed by

$$\begin{aligned} y_0 &= 1 + z_0 + z_1 - b'z_2 - b'z_3, \\ y_1 &= z_1 + z_4 + b'z_5 - z_0 - b'z_6, \\ y_2 &= z_7 - z_2 - z_3 - z_5 - z_6, \\ y_3 &= b'z_0 + z_3 + z_8 - z_2 - b'z_1 - b'z_4, \\ y_4 &= b'z_2 + b'z_3 + b'z_7 - b'z_5 - b'z_6, \\ y_5 &= b'z_3 + b'z_8 - b'z_2, \end{aligned}$$

where $X_0 = x_0 + x_2\rho + x_4\rho^2, X_1 = x_1 + x_3\rho + x_5\rho^2$ and $Y_0 = y_0 + y_2\rho + y_4\rho^2, Y_1 = y_1 + y_3\rho + y_5\rho^2$ $(x_i, y_i \in \mathbb{F}_{3^n})$ for $i = 0, 1, ..., 5$.

Proof. Refer Appendix A. □

From Lemma 2 the proposed algorithm can be obtained. We describe the explicit algorithm of the proposed scheme in Algorithm 3.

Proposition 1. *Algorithm 3 requires $66M_1 + 3(n+1)C_1 + I_3$, where M_1, C_1, I_3 are the cost of multiplication in $\mathbb{F}_{3^n}$, cubing in $\mathbb{F}_{3^n}$, inversion in $\mathbb{F}_{3^{3n}}$, respectively.*

Proof. The computation of $B = A^{3^{3n}-1}$ is as expensive as that of the final exponentiation for the Duursma-Lee algorithm, namely $30M_1 + I_3$ from Eq. (6). The calculations of C and D are performed by a powering to the $(3^n + 1)$-th power. The calculation of E is performed by $(n + 1)/2$ cubings (its cost is $(n+1)/2 \cdot C_6 = 3(n+1)C_1$). We have to calculate E^{-1} in the case of $b' = 1$, which

requires no cost due to Lemma 1. Hence the proposed algorithm of computing the final exponentiation for the η_T pairing needs $(30M_1+I_3)+3(n+1)C_1+2C_T+M_6$, where $C_T = 9M_1$ is the cost of powering to (3^n+1)-th in $T_2(\mathbb{F}_{3^{3n}})$). We thus obtain the cost estimation of this proposition. $\square$.

4.3 How to Apply GPS Encoding to η_T Pairing

In this section we explain how to apply the GPS encoding to the η_T pairing.

The GPS encoding utilizes the arithmetic of the quotient group $\mathcal{G} = \mathbb{F}^*_{3^{6n}}/\mathbb{F}^*_{3^{3n}}$. Note that $\eta_T(P,Q)^V$ for $V = (3^n+1)(3^n+1-b'(3^{(n+1)/2}))$ is contained in $\mathcal{G}$. In order to compute the powering by V we have to compute in $\mathbb{F}_{3^{6n}}$ since $\eta_T(P,Q)$ is contained neither in $\mathcal{G}$ nor $T_2(\mathbb{F}_{3^{3n}})$. We have the relationship $\eta_T(P,Q)^V = CD^{-b'}$, where $C = B^{3^n+1}, D = B^{3^{(n+1)/2}}$, and $B = \eta_T(P,Q)^{3^n+1}$. Algorithm 4 shows the GPS encoding for the η_T pairing.

Algorithm 4: Proposed GPS Encoding of η_T Pairing

input: $A \in \mathbb{F}^*_{3^{6n}}/(\mathbb{F}^*_{3^{6n}})^{3^n+1-b'3^{(n+1)/2}}$
output: GPS encoding of $A \in \mathbb{F}^*_{3^{6n}}$
1. $B \leftarrow A^{3^n+1}$ (in $\mathbb{F}_{3^{6n}}$)
2. $C \leftarrow B^{3^n+1}$ (in $\mathbb{F}_{3^{6n}}$)
3. $D \leftarrow B^{3^{(n+1)/2}}$ (in $\mathbb{F}_{3^{6n}}$)
4. **if** $b' = 1$ **then** $E \leftarrow C \cdot D^{-1}$ (in $\mathbb{F}_{3^{6n}}$)
   else $E \leftarrow C \cdot D$ (in $\mathbb{F}_{3^{6n}}$)
5. **return** E_0/E_1, where $E = E_0 + E_1\sigma$ (in $\mathbb{F}_{3^{3n}}$)

$$\begin{pmatrix} \text{cost: } 90M_1 + 3(n+1)C_1 + 2I_3 & \text{if } b' = 1 \\ 60M_1 + 3(n+1)C_1 + I_3 & \text{if } b' = -1 \end{pmatrix}$$

We estimate the computational cost of Algorithm 4. Recall that $X^{3^n} \in \mathbb{F}_{3^{6n}}$ is computed virtually for free (see [2] or Appendix B). Therefore the cost of computing $X^{3^n+1} = X^{3^n} \cdot X$ is just $M_6 = 18M_1$. The total costs of both Step 1 and 2 are $36M_1$. The cost of $B^{3^{(n+1)/2}}$ is $((n+1)/2) \cdot M_6 = 3(n+1)C_1$. The cost of $C \cdot D^{-1}$ is $M_6 + I_6 = 48M_1 + I_3$ and the cost of $C \cdot D$ is $M_6 = 18M_1$. The computation of E_0/E_1 which is same as the original GPS encoding takes $6M_1 + I_3$. Consequently the GPS encoding for the η_T pairing is $90M_1 + 3(n+1)C_1 + 2I_1$ if $b' = 1$ and $60M_1 + 3(n+1)C_1 + I_1$ if $b' = -1$.

4.4 Comparison

Here we compare the computational cost of the proposed scheme with other schemes.

The computational cost of an exponentiation with cubings and multiplications by bit is $2nM_6/3 + (n-1)C_6 = 12nM_1 + 6(n-1)C_1$ on average. The

Table 2. Comparison of several bilinear pairing algorithms

η_T pairing		computational cost	estimation
Proposed final exponentiation		$\mathbf{66M_1 + (3n + 3)C_1 + I_3}$	$122.8M_1$
(Algorithm 3)	total cost	$(7.5n + 74.5)M_1 + (8n + 8)C_1 + I_3$	$927.1M_1$
Proposed GPS encoding ($b' = 1$)		$\mathbf{90M_1 + (3n + 3)C_1 + 2I_3}$	$162.5M_1$
(Algorithm 4)	total cost	$(7.5n + 98.5)M_1 + (8n + 8)C_1 + 2I_3$	$966.9M_1$
Proposed GPS encoding ($b' = -1$)		$\mathbf{60M_1 + (3n + 3)C_1 + I_3}$	$116.8M_1$
(Algorithm 4)	total cost	$(7.5n + 68.5)M_1 + (8n + 8)C_1 + I_3$	$921.1M_1$
Ordinary final exponentiation		$228M_1 + (3n + 3)C_1 + I_3$	$284.8M_1$
([1, 2])	total cost	$(7.5n + 236.5)M_1 + (8n + 8)C_1 + I_3$	$1089.1M_1$

Duursma-Lee algorithm		computational cost	estimation
Final exponentiation		$30M_1 + I_3$	$45.73M_1$
([9])	total cost	$(15n + 30)M_1 + (10n + 2)C_1 + I_3$	$1636.4M_1$
GPS encoding		$6M_1 + I_3$	$21.73M_1$
([8])	total cost	$(15n + 6)M_1 + (10n + 2)C_1 + I_3$	$1612.4M_1$

We set $n = 97,\ C_1 = 0.1395M_1,\ I_3 = 15.73M_1$ in the estimation.

previously fastest method proposed by [2] for computing the final exponentiation requires $11M_6+(n+1)C_6/2+I_6 = 228M_1+(3n+3)C_1+I_3$, which optimized the number of multiplications and inversions for the final exponentiation without using the torus. We choose the relationship among C_1, M_1, I_3 so

$$C_1 = 0.1395M_1 \text{ and } I_3 = 15.73M_1$$

as described in [8]. Then the cost of the final exponentiation appeared in [2] is $284.8M_1$, which is about 26% of the total cost $1089.1M_1$ of the η_T pairing. Note that there is another relationship among them [7]. A trinomial basis is used in [8], and a normal basis is used in [7] for a basis of $\mathbb{F}_{3^n}$ over $\mathbb{F}_3$. If the normal basis is used, then C_1 becomes virtually for free, however, M_1 becomes considerably higher.

On the other hand, the computational cost of our proposed final exponentiation is $122.8M_1$ and the total takes $927.1M_1$, namely it is about 13% of the whole η_T pairing. Therefore, the proposed scheme can compute the η_T pairing about 17% faster than the previously known algorithms. Table 3 concludes a final exponentiation and the GPS encoding for the η_T pairing.

4.5 Map of η_T Pairing to Tate Pairing

We explain how to efficiently obtain the value of Tate pairing using the output from the variation of the η_T pairing (Algorithm 2).

Raising both sides of Eq. (4) to 3^n-th power yields $(\eta_T(P,Q)^W)^{3^n} = (\eta_T(3^{(n-1)/2}P, Q)^{3^{(n+1)/2}})^W$ due to the bilinearity of $\eta_T(P,Q)^W$. Therefore $(\eta_T(P,Q)^W)^{3^n}$ can be computed by inserting $3^{(n-1)/2}P$ and Q to Algorithm 3.

The calculation of $3^{(n-1)/2}P$ takes the cost of $2(n-1)C_1$ because of $3(x,y) = (x^9 - b, -y^9)$ for $(x,y) \in E^b(\mathbb{F}_{3^n})$. Moreover, 3^n-th root of elements in $\mathbb{F}^*_{3^{6n}}$ is obtained only by computing some additions and subtractions in $\mathbb{F}_{3^n}$ [2] (see Appendix B). Next we consider an efficient map of $\eta_T(P,Q)^W$ to $\hat{e}(P,Q)$. Let $U = \eta_T(P,Q)^W$, then we have the relationship $\hat{e}(P,Q) = U^{-2} \cdot \left(U^{3^{(n+1)/2}} \cdot \sqrt[3^n]{U^{(n-1)/2}}\right)^{-b'}$ [2]. Moreover we set $F = U^2$, $G = U^{3^{(n-1)/3}}$, $H = G^3$, $I = \sqrt[3^n]{G}$. Then $\hat{e}(P,Q) = (FHI)^{-1}$ if $b' = 1$, and $\hat{e}(P,Q) = F^{-1}HI$ otherwise. Consequently we have obtained Algorithm 5 for the efficient map from $\eta_T(P,Q)^W$ to $\hat{e}(P,Q)$. We note that Algorithm 5 uses no inversion.

Algorithm 5: Proposed Map from $\eta_T(P,Q)^W$ to $\hat{e}(P,Q)$

input: $U = \eta_T(P,Q)^W \in \mathbb{F}_{3^{6n}}$
output: $\hat{e}(P,Q) \in \mathbb{F}_{3^{6n}}$
1. $F \leftarrow U^2$ (in $\mathbb{F}_{3^{6n}}$)
2. $G \leftarrow U^{3^{(n+1)/2}}$ (in $\mathbb{F}_{3^{6n}}$)
3. $H \leftarrow G^3$ (in $\mathbb{F}_{3^{6n}}$)
4. $I \leftarrow \sqrt[3^n]{G}$ (in $\mathbb{F}_{3^{6n}}$)
5. **if** $b = 1$ **then**
6. $\quad J \leftarrow F \cdot H \cdot I$ (in $\mathbb{F}_{3^{6n}}$)
7. $\quad$ **return** $J_0 - J_1\sigma$, where $J = J_0 + J_1\sigma, J_0, J_1 \in \mathbb{F}_{3^{3n}}$
8. **else**
9. $\quad J \leftarrow H \cdot I$ (in $\mathbb{F}_{3^{6n}}$)
10. $\quad J \leftarrow (F_0 - F_1\sigma) \cdot J$, (in $\mathbb{F}_{3^{6n}}$) where $F = F_0 + F_1\sigma, F_0, F_1 \in \mathbb{F}_{3^{3n}}$
11. $\quad$ **return** J
12. **end if** (Cost: $54M_1 + (3n+9)C_1$)

The cost of U, $U^{3^{(n+1)/2}}$ and G^3 are $M_6 = 18M_1$, $(n+1)/2 \cdot C_6 = 3(n+1)C_1$, $C_6 = 6C_1$, respectively. Moreover $\sqrt[3^n]{G}$ is computed virtually for free, and the computation of both $F \cdot H \cdot I$ and $(F_0 - F_1\sigma) \cdot H \cdot I$ take $2M_6 = 36M_1$. Consequently the cost of map of $\eta_T(P,Q)^W$ to $\hat{e}(P,Q)$ is $54M_1 + (3n+9)C_1$.

5 Conclusion

In this paper we presented some new efficient algorithms for the final exponentiation of the η_T pairing. We deploy new encoding techniques for the embedded group $\mathbb{F}_{3^{6n}}$, which allow us to efficiently perform the powering by $3^n + 1$ and the inversion. The total cost of computing the η_T pairing with $n = 97$ has become about 17% faster than the previously known methods.

References

1. P. S. L. M. Barreto, S. Galbraith, C. Ó. hÉigeartaigh and M. Scott, "Efficient pairing computation on supersingular abelian varieties," Cryptology ePrint Archive, Report 2004/375, 2004.
2. J-L. Beuchat, M. Shirase, T. Takagi and E. Okamoto, "An algorithm for the η_T pairing calculation in characteristic three and its hardware implementation," Cryptology ePrint Archive, Report 2006/327, 2006.
3. D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," *CRYPTO 2001*, LNCS 2139, pp.213-229, 2001.
4. D. Boneh, C. Gentry and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," *CRYPTO 2005*, LNCS 3621, pp.258-275, 2005.
5. D. Boneh, B. Lynn and H. Shacham, "Short signature from the Weil pairing," *ASIACRYPT 2001*, LNCS 2248, pp.514-532, 2001.
6. I. Duursma and H. S. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$," *ASIACRYPT 2003*, LNCS 2894, pp.111-123, 2003.
7. R. Granger, D. Page and M. Stam, "Hardware and Software Normal Basis Arithmetic for Pairing-Based Cryptography in Characteristic Three," IEEE Transaction on Computers, Vol. 54, No. 7, July 2005, pp.852-860, 2005.
8. R. Granger, D. Page and M. Stam, "On Small Characteristic Algebraic Tori in Pairing-Based Cryptography," Cryptology ePrint Archive, Report 2006/179, 2006.
9. T. Kerins, W. P. Marnane, E. M. Popovici and P. S. L. M. Barreto, "Efficient hardware for the Tate pairing calculation in characteristic three," *CHES 2005*, LNCS 3659, Springer–Verlag, pp.412-426, 2005.
10. S. Kwon, "Efficient Tate pairing computation for supersingular elliptic curves over binary fields," Cryptology ePrint Archive, Report 2004/303, 2004.
11. R. Ronan, C. Ó. hÉigeartaigh, C. Murphy, T. Kerins and P. S. L. M. Baretto, "Hardware Implementation of the η_T Pairing in Characteristic 3," Cryptology ePrint Archive, Report 2006/371, 2006.
12. C. Shu, S. Kwon and K. Gaj, "FPGA Accelerated Tate Pairing Based Cryptosystems over Binary Fields," Cryptology ePrint Archive, Report 2006/179, 2006.

A Proofs of Lemma 2

Lemma 2. *Let $n \equiv 1 \bmod 6$. For $X = X_0 + X_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$ we can compute $Y = \Lambda(X) = X^{3^n+1} = Y_0 + Y_1\sigma$ with 9 multiplications in $\mathbb{F}_{3^n}$ as follows:*
Let $z_0 \sim z_8$ be defined as

$$\begin{array}{lll} z_0 = x_0x_4, & z_1 = x_1x_5, & z_2 = x_2x_4, \\ z_3 = x_3x_5, & z_4 = (x_0 + x_1)(x_4 - x_5), & z_5 = x_1x_2, \\ z_6 = x_0x_3, & z_7 = (x_0 + x_1)(x_2 + x_3), & z_8 = (x_2 + x_3)(x_4 - x_5), \end{array}$$

then, Y can be computed by

$$\begin{aligned} y_0 &= 1 + z_0 + z_1 - b'z_2 - b'z_3, \\ y_1 &= z_1 + z_4 + b'z_5 - z_0 - b'z_6, \\ y_2 &= z_7 - z_2 - z_3 - z_5 - z_6, \\ y_3 &= b'z_0 + z_3 + z_8 - z_2 - b'z_1 - b'z_4, \\ y_4 &= b'z_2 + b'z_3 + b'z_7 - b'z_5 - b'z_6, \\ y_5 &= b'z_3 + b'z_8 - b'z_2, \end{aligned}$$

where $X_0 = x_0+x_2\rho+x_4\rho^2, X_1 = x_1+x_3\rho+x_5\rho^2$ *and* $Y_0 = y_0+y_2\rho+y_4\rho^2, Y_1 = y_1+y_3\rho+y_5\rho^2$ $(x_i, y_i \in \mathbb{F}_{3^n})$ *for* $i = 0, 1, ..., 5$.

Proof. Note that $\rho^{3^n} = \rho + b'$, $(\rho^2)^{3^n} = \rho^2 - b'\rho + 1$ in the case of $n \equiv 1 \bmod 6$. For $X = X_0 + X_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$ we have the relationship:

$$X^{3^n+1} = X_0^{3^n+1} + X_1^{3^n+1} + (X_0^{3^n}X_1 - X_1^{3^n}X_0)\sigma,$$

$$X_0^{3^n} = (x_0 + b'x_2 + x_4) + (x_2 - b'x_4)\rho + x_4\rho^2, \tag{7}$$

$$\begin{aligned} X_0^{3^n+1} = (x_0^2 + b'x_0x_2 + x_0x_4 - b'x_2x_4 - x_4^2) + (-x_0x_2 - b'x_0x_4 + b'x_2^2)\rho \\ +(-x_0x_4 + x_2^2 - x_4^2)\rho^2. \end{aligned} \tag{8}$$

We similarly see

$$X_1^{3^n} = (x_1 + b'x_3 + x_5) + (x_3 - b'x_5)\rho + x_5\rho^2, \tag{9}$$

$$\begin{aligned} X_1^{3^n+1} = (x_1^2 + b'x_1x_3 + x_1x_5 - b'x_3x_5 - x_5^2) + (-x_1x_3 - b'x_1x_5 + b'x_3^2)\rho \\ +(-x_1x_5 + x_3^2 - x_5^2)\rho^2. \end{aligned} \tag{10}$$

$X_0^2 + X_1^2 = 1$ is satisfied since $X = X_0 + X_1\sigma \in T_2(\mathbb{F}_{3^{3n}})$. This derives the following equation.

$$(x_0 + x_2\rho + x_4\rho^2)^2 + (x_1 + x_3\rho + x_5\rho^2)^2 = 1 \ (= 1 + 0\rho + 0\rho^2).$$

This equation gives

$$\begin{cases} x_0^2 + x_1^2 = 1 + b'x_2x_4 + b'x_3x_5 \\ x_2^2 + x_3^2 = x_0x_4 + x_1x_5 - b'x_0x_2 - b'x_1x_3 - b'x_2x_4 - b'x_3x_5 \\ x_4^2 + x_5^2 = b'x_0x_2 + b'x_1x_3 + b'x_2x_4 + b'x_3x_5. \end{cases} \tag{11}$$

By Eqs. (8), (10), (11)

$$\begin{aligned} X_0^{3^n+1} + X_1^{3^n+1} &= y_0 + y_2\rho + y_4\rho^2 \\ &= (1 + x_0x_4 + x_1x_5 - b'x_2x_4 - b'x_3x_5) \\ &\quad +(x_0x_2 + x_1x_3 - x_2x_4 - x_3x_5)\rho \\ &\quad +(b'x_0x_2 + b'x_1x_3 + b'x_2x_4 + b'x_3x_5)\rho^2. \end{aligned}$$

And by Eqs. (7), (9), (11)

$$\begin{aligned} X_0^{3^n}X_1 - X_1^{3^n}X_0 &= y_1 + y_3\rho + y_5\rho^2 \\ &= (b'x_1x_2 + x_1x_4 - b'x_0x_3 - x_0x_5) \\ &\quad +(b'x_0x_5 + x_3x_4 - b'x_1x_4 - x_2x_5)\rho \\ &\quad +(b'x_3x_4 - b'x_2x_5)\rho^2. \end{aligned}$$

Moreover, we define z_0, ... , z_8 by

$$\begin{gathered} z_0 = x_0x_4,\ z_1 = x_1x_5,\ z_2 = x_2x_4,\ z_3 = x_3x_5,\ z_4 = (x_0 + x_1)(x_4 - x_5), \\ z_5 = x_1x_2,\ z_6 = x_0x_3,\ z_7 = (x_0 + x_1)(x_2 + x_3),\ z_8 = (x_2 + x_3)(x_4 - x_5). \end{gathered}$$

Then we have

$$\begin{aligned}
y_0 &= 1 + x_0x_4 + x_1x_5 - b'x_2x_4 - b'x_3x_5 &&= 1 + z_0 + z_1 - b'z_2 - b'z_3,\\
y_1 &= b'x_1x_2 + x_1x_4 - b'x_0x_3 - x_0x_5 &&= z_1 + z_4 + b'z_5 - z_0 - b'z_6,\\
y_2 &= x_0x_2 + x_1x_3 - x_2x_4 - x_3x_5 &&= z_7 - z_2 - z_3 - z_5 - z_6,\\
y_3 &= b'x_0x_5 + x_3x_4 - b'x_1x_4 - x_2x_5 &&= b'z_0 + z_3 + z_8 - z_2 - b'z_1 - b'z_4,\\
y_4 &= b'x_0x_2 + b'x_1x_3 + b'x_2x_4 + b'x_3x_5 &&= b'z_2 + b'z_3 + b'z_7 - b'z_5 - b'z_6,\\
y_5 &= b'z_3z_4 - b'z_2z_5 &&= b'z_3 + b'z_8 - b'z_2.
\end{aligned}$$

Consequently Lemma 2 is showed. □

B Powering by 3^n and 3^n-th Root in $\mathbb{F}_{3^n}$

This section explains that a powering by 3^n or 3^n-th root in $\mathbb{F}_{3^{6n}}$ is computed virtually for free. It follows that

$$\sigma^{3^n} = \begin{cases} \sigma & \text{if } n \equiv 0 \pmod 2 \\ -\sigma & \text{if } n \equiv 1 \pmod 2 \end{cases}, \quad \rho^{3^n} = \begin{cases} \rho & \text{if } n \equiv 0 \pmod 3 \\ \rho + b' & \text{if } n \equiv 1 \pmod 3 \\ \rho - b' & \text{if } n \equiv 2 \pmod 3 \end{cases}.$$

Suppose that $n \equiv 1 \pmod 6$ which is a necessary condition of the Duursma-Lee algorithm and the η_T pairing. Then we have

$$\sigma^{3^n} = -\sigma,\ \rho^{3^n} = \rho + b',\ (\rho^2)^{3^n} = \rho^2 - b'\rho + 1.$$

B.1 Powering by 3^n

Let $Y = X^{3^n}$ for $X \in \mathbb{F}^*_{3^{6n}}$, where $X = x_0 + x_1\sigma + x_2\rho + x_3\sigma\rho + x_4\rho^2 + x_5\sigma\rho^2$ and $Y = y_0 + y_1\sigma + y_2\rho + y_3\sigma\rho + y_4\rho^2 + y_5\sigma\rho^2$ for some $x_i,\ y_i \in \mathbb{F}_{3^n}$. Then we have

$$\begin{cases} y_0 = x_0 + b'x_2 + x_4 \\ y_1 = -x_1 - b'x_3 - x_5 \\ y_2 = x_2 - b'x_4 \\ y_3 = -x_3 + b'x_5 \\ y_4 = x_4 \\ y_5 = -x_5 \end{cases} \tag{12}$$

since

$$\begin{aligned}
&(x_0 + x_1\sigma + x_2\rho + x_3\sigma\rho + x_4\rho^2 + x_5\sigma\rho^2)^{3^n}\\
&= x_0 + x_1(\sigma)^{3^n} + x_2(\rho)^{3^n} + x_3(\sigma\rho)^{3^n} + x_4(\rho^2)^{3^n} + x_5(\sigma\rho^2)^{3^n}\\
&= x_0 + x_1(-\sigma)^{3^n} + x_2(\rho + b') + x_3(-\sigma\rho - b'\sigma) + x_4(\rho^2 - b'\rho + 1)\\
&\qquad + x_5(\sigma\rho^2 + b'\sigma\rho - \sigma)\\
&= (x_0 + b'x_2 + x_4) + (-x_1 - b'x_3 - x_5)\sigma + (x_2 - b'x_4)\rho\\
&\qquad + (-x_3 + b'x_5)\sigma\rho + x_4\rho^2 - x_5\sigma\rho^2.
\end{aligned}$$

Note that $x_i^{3^n} = x_i$ and $y_i^{3^n} = y_i$ since $x_i,\ y_i \in \mathbb{F}_{3^n}$. Therefore a powering by 3^n is computed virtually for free.

B.2 3^n-th root

Let $Y = \sqrt[3^n]{X}$ for $X \in \mathbb{F}_{3^{6n}}^*$, where $X = x_0 + x_1\sigma + x_2\rho + x_3\sigma\rho + x_4\rho^2 + x_5\sigma\rho^2$ and $Y = y_0 + y_1\sigma + y_2\rho + y_3\sigma\rho + y_4\rho^2 + y_5\sigma\rho^2$ for some $x_i,\ y_i \in \mathbb{F}_{3^n}$. Note that a 3^n-th root operation in characteristic three is uniquely determined. We have

$$\begin{cases} x_0 = y_0 + b'y_2 + y_4 \\ x_1 = -y_1 - b'y_3 - y_5 \\ x_2 = y_2 - b'y_4 \\ x_3 = -y_3 + b'y_5 \\ x_4 = y_4 \\ x_5 = -y_5 \end{cases} \tag{13}$$

by Eq.(12) since $X = Y^{3^n}$. Solving Eq.(13) for each y_i gives

$$\begin{cases} y_0 = x_0 - b'x_2 + x_4 \\ y_1 = -x_1 + b'x_3 - x_5 \\ y_2 = x_2 + b'x_4 \\ y_3 = -x_3 - b'x_5 \\ y_4 = x_4 \\ y_5 = -x_5. \end{cases}$$

Therefore a 3^n-th root operation is computed virtually for free.