Memoirs of the Faculty of Engineering, Okayama University, Vol. 44, pp. 69-72, January 2010

Efficient Squaring Algorithm for Xate Pairing with Freeman Curve

Kenta NEKADO*
Graduate School of Natural Science and Technology, Okayama University
3-1-1, Tsushima-naka, Okayama, Okayama 700-8530, Japan

Hidehiro KATO*
Graduate School of Natural Science and Technology, Okayama University
3-1-1, Tsushima-naka, Okayama, Okayama 700-8530, Japan

Yasuyuki NOGAMI*
Graduate School of Natural Science and Technology, Okayama University
3-1-1, Tsushima-naka, Okayama, Okayama 700-8530, Japan

Yoshitaka MORIKAWA*
Graduate School of Natural Science and Technology, Okayama University
3-1-1, Tsushima-naka, Okayama, Okayama 700-8530, Japan

(Received December 6, 2009)

Recently, pairing–based cryptographies have attracted much attention. For fast pairing calculation, not only pairing algorithms but also arithmetic operations in extension field should be efficient. Especially for final exponentiation included in pairing calculation, squaring is more important than multiplication. This paper proposes an efficient squaring algorithm in extension field for Freeman curve.

1 INTRODUCTION

In recent years, pairing–based cryptographies such as ID–based cryptography [1] and group signature [2] have attracted much attention. For their implementations, pairings such as Weil pairing [1], Tate pairing, Ate pairing [3] and Xate pairing [4] can be efficiently applied. In order to implement these pairings, several kinds of ordinary pairing–friendly curves such as Miyaji–Nakabayashi–Takano (MNT) curve [5], Barreto–Naehrig (BN) curve [6] and Freeman curve [7], [8] have been proposed. As the definition field of these curves, most of researchers use optimal extension field (OEF) [9] because OEF carries out arithmetic operations efficiently. However, it is known that OEF is not available for the definition field of Freeman curve due to the mismatch of some conditions [10]. On the other hand, Kato ea al. have proposed type–X all one polynomial field (AOPF) [11, 12]. It can carry out arithmetic operations as efficient as OEF, and is available for the definition field of Freeman curve.

As our previous work [10], the authors have considered how to construct type–X AOPF for Xate pairing with Freeman curve and optimized the multiplication algorithm. However, especially for final exponentiation included in Xate pairing calculation, squarings are more important than multiplications. Thus, this paper proposes an efficient squaring algorithm in the type–X AOPF. Then, it is shown that the proposed algorithm makes a squaring about 10 percent faster than the conventional algorithm.

*E-mail: { nekado, kato, nogami, morikawa }@trans.cne.okayama-u.ac.jp

Notation: $\mathbb{F}_p$, $\mathbb{F}_{p^m}$, $\mathbb{F}^*_{p^m}$, and $E(\mathbb{F}_{p^m})$ denote a prime field, an m–th extension field over $\mathbb{F}_p$, the multiplicative group in $\mathbb{F}_{p^m}$, and the elliptic curve defined over $\mathbb{F}_{p^m}$. For two integers m and n, $m \,|\, n$ means that m divides n. M_m, S_m, A_m, and D_m denote the computational costs of a multiplication, a squaring, an addition (a subtraction), and a doubling in $\mathbb{F}_{p^m}$, respectively.

2 FUNDAMENTALS

This section runs over Freeman curve, Xate pairing, type–X all one polynomial field (AOPF), and efficient multiplication and squaring algorithms in type–X AOPF.

2.1 Xate pairing with Freeman curve

The smallest positive integer d such that $r \,|\, (p^d - 1)$ is called *embedding degree*, where r is the group order for pairing.

Freeman curve is a class of ordinary pairing–friendly curves of embedding degree $d = 10$ [7], [8]. The characteristic and order of Freeman curve $E(\mathbb{F}_{p^m})$ are given as

$$p(\chi) = 25\chi^4 + 25\chi^3 + 25\chi^2 + 10\chi + 3, \tag{1a}$$
$$r(\chi) = 25\chi^4 + 25\chi^3 + 15\chi^2 + 5\chi + 1, \tag{1b}$$

where χ is an integer such that $p(\chi)$ becomes a prime number.

Nogami et al. have proposed integer χ–based (Xate) pairing [4]. In this paper, the authors focus on Xate pairing with Freeman curve. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be

$$\mathbb{G}_1 = E(\mathbb{F}_{p^d})[r] \cap \mathrm{Ker}(\phi - [1]), \tag{2a}$$
$$\mathbb{G}_2 = E(\mathbb{F}_{p^d})[r] \cap \mathrm{Ker}(\phi - [p]), \tag{2b}$$

where $E(\mathbb{F}_{p^d})[r]$ denotes the set of rational points of order r in $E(\mathbb{F}_{p^d})$. Let $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, in the case of Freeman curve, Xate pairing ϵ is given as

$$\epsilon : \begin{cases} \mathbb{G}_2 \times \mathbb{G}_1 & \to \ \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r, \\ (Q,\, P) & \mapsto \ \hat{f}_{\chi,Q}(P)^{(p^{10}-1)/r}. \end{cases} \tag{3a}$$

$$\hat{f}_{\chi,Q}(P) = \left(f_{\chi,Q}(P)^{(1+p)} \cdot g_{\chi Q,\, p\chi Q}\right)^{1+p^3} \cdot g_{\chi Q + p\chi Q,\, p^3(\chi Q + p\chi Q)}. \tag{3b}$$

where g_{Q_1, Q_2} denotes the line passing through two points Q_1, Q_2. It gives a non–degenerate and bilinear map. Xate pairing consists of two principal steps, one is $f_{\chi,Q}(P)$ calculation by Miller's algorithm, and the other is the calculation called *final exponentiation* that $\hat{f}_{\chi,Q}(P)$ is raised to the $((p^d-1)/r)$–th power.

Additionally, Nogami et al. have improved Xate pairing by using subfield–twisted curve. It is called cross–twisted Xate (Xt–Xate) pairing [4]. In the case of Freeman curve $E(\mathbb{F}_{p^{10}})$, we can use *quadratic twisted* curve $E'(\mathbb{F}_{p^5})$ as the subfield–twisted curve, for which we need to prepare subfield $\mathbb{F}_{p^5}$ besides the definition field $\mathbb{F}_{p^{10}}$.

2.2 Type–X All One Polynomial Field

Kato ea al. have proposed type–I eXtended all one polynomial field (type I–X AOPF) [11] and type–II eXtended AOPF (type II–X AOPF) [12]. This paper calls them type–X AOPF collectively. Type–X AOPF $\mathbb{F}_{(p^n)^m}$ is constructed by m–th towering over $\mathbb{F}_{p^n}$ with a speciall class of type–$\langle k, m\rangle$ Gauss period normal bases (GNBs) [13] when $\gcd(m,n)=1$. Type–$\langle k, m\rangle$ GNB is defined with a certain integer k as follows.

Define 1 : Let $km+1$ be a prime number not equal to p. Suppose that $\gcd(km/e, m) = 1$, where e is the order of p in $\mathbb{F}_{km+1}$. Then, for any primitive k–th root θ of unity in $\mathbb{F}_{km+1}$ and primitive $(km+1)$–st root β of unity in $\mathbb{F}_{(p^n)^e}$,

$$\gamma = \sum_{i=0}^{k-1} \beta^i \in \mathbb{F}_{(p^n)^m} \tag{4}$$

generates a normal basis $\{\gamma, \gamma^p, \cdots, \gamma^{p^{m-1}}\}$ in $\mathbb{F}_{(p^n)^m}$. It is called type–$\langle k, m\rangle$ GNB. ■

There exists a special class of type–$\langle k, m\rangle$ GNBs of type–X AOPF for every pair of characteristic p and extension degree m when $p > m$ [11, 12]. Thus, type–X AOPF is available for the definition field of Freeman curve. For example, we can prepare the subfield $\mathbb{F}_{p^5}$ by 5–th exteding over $\mathbb{F}_p$ with the type–$\langle k_1, m=5\rangle$ GNB, and the definition field $\mathbb{F}_{p^{10}}$ as type–X AOPF $\mathbb{F}_{(p^5)^2}$ by 2–nd towering over the $\mathbb{F}_{p^5}$ with the type–$\langle k_2, m=2\rangle$ GNB as shown in **Fig**.1. In what follows, in order to make squarings in this $\mathbb{F}_{(p^5)^2}$ more efficient, we consider an efficient squaring algorithm in type–X AOPF $\mathbb{F}_{(p^n)^2}$.

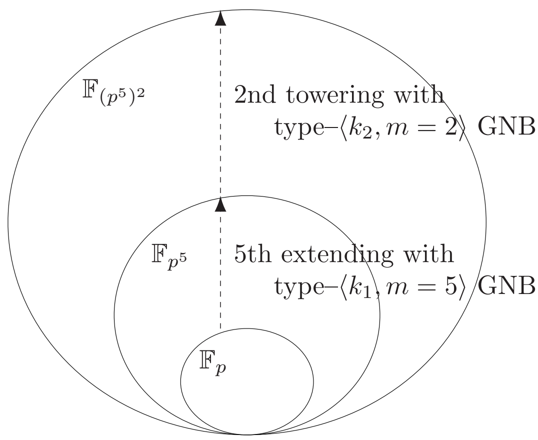

Fig 1: Type–X AOPF $\mathbb{F}_{(p^5)^2}$

2.3 Cyclic Vector Multiplication Algorithm

As an efficient multiplication algorithm in type–X AOPF, Kato et al. have proposed *cyclic vector multiplication algorithm* (CVMA) [11, 12]. This subsection shows CVMA in type–X AOPF $\mathbb{F}_{(p^n)^2}$.

Let X, Y, $Z \in \mathbb{F}_{(p^n)^2}$ be

$$X = x_0\gamma + x_1\gamma^p, \qquad x_0, x_1 \in \mathbb{F}_{p^n}, \tag{5a}$$
$$Y = y_0\gamma + y_1\gamma^p, \qquad y_0, y_1 \in \mathbb{F}_{p^n}, \tag{5b}$$
$$Z = XY = z_0\gamma + z_1\gamma^p, \qquad z_0, z_1 \in \mathbb{F}_{p^n}, \tag{5c}$$

and k' be

$$k' = \begin{cases} (k+1)/2 & \text{(when } k \text{ is odd)} \\ -k/2 & \text{(when } k \text{ is even)} \end{cases}, \tag{6}$$

then CVMA calculates a multiplication in $\mathbb{F}_{(p^n)^2}$ as follows.

$$z_0 = k'(x_0 - x_1)(y_0 - y_1) - x_0 y_0, \tag{7a}$$
$$z_1 = k'(x_0 - x_1)(y_0 - y_1) - x_1 y_1. \tag{7b}$$

Its calculation cost is given as

$$M_{2n} = 3M_n + 4A_n + K_n, \tag{8}$$

where K_n is the computational cost of a scalar k' multiplication in $\mathbb{F}_{p^n}$.

On the other hand, let X, $Z \in \mathbb{F}_{(p^n)^2}$ be

$$X = x_0\gamma + x_1\gamma^p, \qquad x_0, x_1 \in \mathbb{F}_{p^n}, \tag{9a}$$
$$Z = X^2 = z_0\gamma + z_1\gamma^p, \qquad z_0, z_1 \in \mathbb{F}_{p^n}, \tag{9b}$$

then CVMA calculates a squaring in $\mathbb{F}_{(p^n)^2}$ as follows.

$$z_0 = k'(x_0 - x_1)^2 - x_0^2, \tag{10a}$$
$$z_1 = k'(x_0 - x_1)^2 - x_1^2. \tag{10b}$$

Its calculation cost is given as

$$S_{2n} = 3S_n + 3A_n + K_n. \tag{11}$$

When $k = 1$ or 2, K_n of Eqs.(8) and (11) becomes 0. Therefore, in these cases, multiplications and squarings with CVMA are the most efficient, respectively.

2.4 Efficient Squaring Algorithm When $k = 1$

Kato et al. have proposed an efficient squaring algorithm in type–X AOPF constructed by type–$\langle k = 1, m\rangle$ GNB. It is based on the idea as

$$A^2 - B^2 = (A - B)(A + B). \tag{12}$$

For example, the algorithm calculates a squaring in $\mathbb{F}_{(p^n)^2}$ as

$$z_0 = -x_1\{(x_0 - x_1) + x_0\}, \tag{13a}$$
$$z_1 = x_0\{(x_0 - x_1) - x_1\}, \tag{13b}$$

then its calculation cost is given as

$$S_{2n} = 2M_n + 3A_n. \tag{14}$$

Thus, when $M_n/S_n < 1.5$, this algorithm makes squarings more efficient.

However, type–$\langle k = 1, m = 2\rangle$ GNB exists in only 50 percent for every characteristic p. Additionally, as shown in [10], type–$\langle k = 1, m = 2\rangle$ GNB can not be constructed the definition field $\mathbb{F}_{(p^5)^2}$ for the four kinds of Freeman curves shown in [7, 8] although type–$\langle k = 2, m = 2\rangle$ GNB can. Thus, we need an efficient squaring algorithm in the case of the other type–$\langle k, m = 2\rangle$ GNBs.

3 PROPOSED ALGORITHM

This section proposes an efficient squaring algorithm in type–X AOPF constructed by type–$\langle k, m = 2\rangle$ GNB.

3.1 Efficient Squaring Algorithm When $k \neq 1$

Let U and V in $\mathbb{F}_{(p^n)^2}$ be

$$U = u\gamma + u\gamma^p, \qquad V = v\gamma - v\gamma^p, \tag{15a}$$
$$u = (x_0{+}x_1)/2, \qquad v = (x_0{-}x_1)/2. \tag{15b}$$

With U and V, X of Eq.(9a) is written as

$$X = U + V = x_0\gamma + x_1\gamma^p, \tag{16a}$$
$$x_0 = u + v, \quad x_1 = u - v. \tag{16b}$$

then Z of Eq.(9b) is given as

$$Z = X^2 = U^2 + V^2 + 2UV = z_0\gamma + z_1\gamma^p. \tag{17}$$

U^2, V^2 and UV is calculated by using Eqs.(7), (10) as

$$U^2 = -u^2\gamma - u^2\gamma^p, \tag{18a}$$
$$V^2 = (4k' - 1)v^2\gamma + (4k' - 1)v^2\gamma^p, \tag{18b}$$
$$UV = -uv\gamma + uv\gamma^p. \tag{18c}$$

Thus, z_0 and z_1 of Eq.(10) are given as

$$\begin{aligned} z_0 &= -u^2 + (4k' - 1)v^2 - 2uv \\ &= -(u + v)\{(u + v) - 4k'v\} - 4k'uv, \end{aligned} \tag{19a}$$
$$\begin{aligned} z_1 &= -u^2 + (4k' - 1)v^2 + 2uv \\ &= -(u + v)\{(u + v) - 4k'v\} - 4(k' - 1)uv, \end{aligned} \tag{19b}$$

then they are calculated by using Eq.(15b) as

$$\begin{aligned} z_0 = &- x_0\{x_0 + 2k'(x_0 - x_1)\} \\ &- k'(x_0 + x_1)(x_0 - x_1), \end{aligned} \tag{20a}$$
$$\begin{aligned} z_1 = &- x_0\{x_0 + 2k'(x_0 - x_1)\} \\ &- (k' - 1)(x_0 + x_1)(x_0 - x_1). \end{aligned} \tag{20b}$$

When Eq.(20) is calculated with the algorithm as **Fig**.2,

1. $a_0 \leftarrow x_0 + x_1, \quad a_1 \leftarrow x_0 - x_1.$
2. $a_2 \leftarrow x_0 + 2k'a_1.$
3. $b_0 \leftarrow x_0a_2, \quad b_1 \leftarrow a_0a_1.$
4. $z_0 \leftarrow -b_0 - k'b_1, \quad z_1 \leftarrow z_0 - b_1.$

(End of algorithm)

Fig 2: The improved squaring algorithm in $\mathbb{F}_{(p^n)^2}$

the computation amount of a squaring in $\mathbb{F}_{(p^n)^2}$ is given as

$$S_{2n} = 2M_n + 5A_n + K_n + K_n^{(2)}, \tag{21}$$

where $K_n^{(2)}$ is the computational cost of a scalar $2k'$ multiplication in $\mathbb{F}_{p^n}$. Thus, when M_n and $S_n >> A_n$ and $K_n^{(2)}$, this algorithm often makes squarings more efficient than CVMA. Additionally, when $k = 2$ then this algorithm is the most efficient because K_n and $K_n^{(2)}$ become 0. On the other handd, when $k = 1$ then the algorithm of **Sec**.2.4 is more efficient than this algorithm.

3.2 Efficiency for Freeman Curve

The CVMA optimized in [10] makes squarings in both the $\mathbb{F}_{p^5}$ and $\mathbb{F}_{(p^5)^2}$ more efficient than original CVMA. Moreover, the squaring algorithm of the previous subsection converts the computation amounts of a squaring in the $\mathbb{F}_{(p^5)^2}$ as shown in **Table** 1, where "original", "1st improved" or "2nd improved" denote squarings with original CVMA, the CVMA optimized in [10], and the algorithm of the previous subsection in addition to "1st improved" algorithm, respectively.

Table 1: The computation amounts of a squaring

	original	1–st improved	2–nd imporoved
$\mathbb{F}_{p^5}$	(15, 70, 0) †	(15, 40, 10) †	—
$\mathbb{F}_{(p^5)^2}$	(45, 230, 0) †	(45, 140, 30) †	(30, 125, 25) †

† For example, (15, 40, 10) denotes $2M_1 + 3A_1 + 4D_1$ $(S_1 = M_1)$.

The next section concretely shows the efficiency of the proposed algorithm for Freeman curve.

4 EXPERIMENTAL RESULTS

This experiment used Freeman curve with 196–bit characteristic the same as in [10]. **Table** 2 shows the calculation timings of a squaring in each $\mathbb{F}_{p^5}$ and $\mathbb{F}_{(p^5)^2}$ with the computational environment **Table** 3. As shown in **Table** 2, the proposed algorithm makes squarings about 10 percent faster.

Table 2: The calculation timings of a squaring

	original	1–st improved	2–nd improved
$\mathbb{F}_{p^5}$	8.04 μs	7.03 μs	—
$\mathbb{F}_{(p^5)^2}$	20.3 μs	18.1 μs	16.3 μs

Table 3: Computational environment

CPU	Pentium4 3.00GHz
Cache Size	512 KB
OS	Linux 2.6.27
Language	C
Compiler	gcc 4.3.1
Library	GNU MP 4.2.4 [15]

Finally, **Table** 4 shows the experimental result of Xt–Xate pairing with Freeman curve. As shown in **Table** 4, the proposed squaring algorithm is more efficient for Xt–Xate paring with Freeman curve.

Table 4: The calculation times of Xt-Xate pairing

	original	1–st improved	2–nd improved
Miller's algorithm	7.74 ms	6.94 ms	6.73 ms
Final exponentiation	4.28 ms	3.79 ms	3.53 ms
Total	12.0 ms	10.7 ms	10.3 ms

REFERENCES

[1] R. Sakai, K. Ohgishi and M. Kasahara: SCIS 2000, (Jan. 2000), Okinawa, 26–28.

[2] D. Boneh, X. Boyan and H. Shacham: Proc. of Crypto2004, Lect. Notes Comput. Sci., 3152 (2004), 41–55.

[3] H. Hess, N. P. Smart and F. Vercauteren, "The eta pairing re-visited," IEEE Trans. Inf. Theory, Vol. 52, pp. 4595-4602 (2006).

[4] Y.Nogami, M.Akane, Y.Sakemi, H.Kato and Y.Morikawa: Pairing 2008, LNCS, 5209 (Aug. 2008), 178–191.

[5] A. Miyaji, M. Nakabayashi and S. Takano: IEICE Trans. Fundam., E84–A(5) (2001), 1234–1243.

[6] A. J. Devegili, M. Scott and R. Dahab: LNCS, 4575 (2007), 197–207.

[7] D. Freeman: ePrint (2006).

[8] D. Freeman: LNCS, Springer Berlin/Heidelberg, 4076 (Oct. 2006), 452–465.

[9] H. Cohen and G. Frey, "Handbook of elliptic and hyperelliptic curve cryptography," Chapman & Hall/CRC (2005).

[10] K. Nekado, H. Kato, Y. Nogami, Y. Morikawa: Memoirs Faculty Eng. Okayama Univ., 43–14 (2008), pp. 107–112 .

[11] H. Kato, Y. Nogami, T. Yoshida and Y. Morikawa: ETRI Journal (Dec. 2007), 29–6 .

[12] H. Kato, Y. Nogami, T. Yoshida and Y. Morikawa: IEICE Trans. Fundam., E92–A(1) (Jan. 2009), 173–181 .

[13] S. Gao: Doctoral thesis (1993), Waterloo, Ontario, Canada.

[14] H. Kato, Y. Nogami and Y. Morikawa: ITC–CSCC2008 (Jul. 2008), 273–275.

[15] GNU Multiple Precision Arithmetic Library, available at "http://gmplib.org".