SCIS 2008 The 2008 Symposium on Cryptography and Information Security
Miyazaki, Japan, Jan. 22-25, 2008
The Institute of Electronics, Information and Communication Engineers

Efficient $GF(3^m)$ Multiplication Algorithm for ηT Pairing

Gen Takahashi * Fumitaka Hoshino * Tetutaro Kobayashi *

Abstract— The computation speed of pairing based cryptosystems is slow compared with the other public key cryptosystems even though several efficient computation algorithms have been proposed. Thus more efficient computation of the Tate pairing is an important research goal. $GF(3^m)$ multiplication in $GF(3^{6m})$ in the pairing algorithm is the greatest consumer of time. Past research concentrated on reducing the number of $GF(3^m)$ multiplications, for instance the Karatsuba method. In this article, we propose a new method to reduce the number of online precomputations (precomputations) in $GF(3^m)$ multiplications for the ηT pairing. The proposed algorithm reduces 18 online precomputations in $GF(3^{6m})$ in the ηT pairing to 4 online precomputations by reusing the intermediate products obtained in precomputation. We implement the proposed algorithm and compare the time taken by the proposed algorithm with that of the previous work. Our algorithm offers a 40% performance increase for $GF(3^m)$ multiplications in $GF(3^{6m})$ on an AMD 64-bit processor. Additionally, a completely new finding is obtained. The results show that the reducing the number of the multiplications in $GF(3^{6m})$ does not necessarily lead to a speed-up of the ηT pairing calculation.

Keywords: ηT Pairing, $GF(3^m)$, Multiplication, ID-Based cryptosystems, public key cryptosystems

1 Introduction

A lot of cryptosystems based on pairing have been proposed recently such as ID-Based cryptosystems[1] and short signatures[2]. The pairing algorithm is composed of the miller's algorithm and final exponentiation. Recently, Duursma-Lee[6] introduced a method of computing the miller's algorithm on the elliptic curve defined over $GF(3^m)$. The ηT pairing, which reduces the number of the loops in the miller's algorithm, was proposed by Barreto et al[7]. Shirase et al. presented a more efficient variation of final exponentiation[8]. Kawahara et al. introduced an algorithm that efficiently uses registers and implemented the ηT pairing on an AMD 64-bit processor in 2007; the computation timing is 412.88 microseconds (μs)[14]. The computation speed of pairing based cryptosystems is slow compared with the other public key cryptosystems even though several efficient computation algorithms have been proposed.

The computation time of the modified miller's algorithm is three times that of final exponentiation in these studies. Algorithm 1 summarizes the ηT pairing algorithm using the tower fields proposed by Barreto et al. The modified miller's algorithm requires computations in $GF(3^{6m})$, denoted fg, cubings, and cube rootings. Cubing and cube rooting are implemented using frobenius maps. Computations in $GF(3^{6m})$ implemented using the tower of extensions[4] require 18 $GF(3^m)$ multiplications according to the Schoolbook method. Computations in $GF(3^{6m})$ are dominant in the ηT pairing with regard to computation time because computation of frobenius maps only requires memory operations and reduction. Thus, speeding up the computations in $GF(3^{6m})$ is important for the efficient computation of the ηT pairing.

algorithm 1 Computation of ηT Pairing
on $E(GF(3^m))$: $y^2 = x^3 - x - 1, m \pm mod12$
INPUT: $P = (x_p, y_p), Q = (x_q, y_q) \in E(GF(3^m))$
OUTPUT: $\eta T Paring(P, Q) \in GF(3^{6m})$

1. $y_p \leftarrow -y_p$
2. $f \leftarrow -y_p(x_p + x_q + 1) + y_q\sigma + y_p\rho$
3. $for\ \ i \leftarrow 0\ \ to\ \ (m-1)/2\ \ do$
4. $\quad u \leftarrow x_p + x_q + 1$
5. $\quad g \leftarrow -u^2 + y_p y_q \sigma - u\rho - \rho^2$
6. $\quad f \leftarrow fg$ ($GF(3^{6m})$ multiplication)
7. $\quad y_p \leftarrow y_p^{1/3}, y_q \leftarrow y_q^{1/3}$
8. $\quad x_q \leftarrow x_q^3, y_q \leftarrow y_q^3$
9. end for
10. return $f^{(3^{3m}-1)(3^m+1)(3^m-3^{(m+1)/2}+1)}$

Regarding the speeding up of a computation, several methods that reduce the number of multiplications have been proposed, for intance the Karatsuba[3] method shown in Section 2 other than the above method. This article proposes an efficient multiplication algorithm and scheduling in $GF(3^{6m})$ for ηT pairing. A multiplication is composed of an online precomputation (precomputation) and a multiplication of polyno-

* Information Sharing Platform Laboratories NTT 3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan

mials. One multiplication of $GF(3^m)$ in $GF(3^{6m})$ in the ηT pairing requires one precomputation and one multiplication of polynomials. The precomputed intermediate products are obtained from the multiplier of $GF(3^m)$. In subsequent multiplications, if the multiplier is the same, the intermediate products can be reused. The proposed algorithm reduces the precomputations in $GF(3^{6m})$ in the ηT pairing by reusing intermediate products. As a result, 18 precomputations reduce to 4 precomputations in the ηT pairing. We implement the proposed algorithm on a 64-bit processor and compare the time taken by the proposed algorithm with that of previous work. Our algorithm shortens the computation time of $GF(3^m)$ multiplications in $GF(3^{6m})$ in the ηT pairing by 40 %.

In Section 2 we show the ηT pairing algorithm and previous multiplication scheduling approach for the ηT pairing. Second, we describe our idea of reusing intermediate products and the result of implementation in Sections 3 and 4. Section 5 provides the reader with a comparison against previous results.

2 Previous Multiplication Algorithms for ηT the Pairing Calculation

In this section, we decribe 2 $GF(3^m)$ multiplication scheduling approaches that use the tower of extensions for reducing the number of $GF(3^m)$ multiplications in $GF(3^{6m})$; The number of $GF(3^m)$ multiplications is smaller with the Karatsuba methods than with the Schoolbook method.

Calculations in $GF(3^{6m})$ are implemented using the tower of extensions. In the tower of extensions, $GF(3^m)$ extends to $GF(3^{3m})$ and $GF(3^{3m})$ extends to $GF(3^{6m})$. Let $A0, A1, B0, B1, C_0, C_1 be \in GF(3^{3m})$ in Algorithm 1; the symbol fg indicates $C_0 + C_1\sigma + C_1\sigma^2 = (A_0 + A_1\sigma) \times (B_0 + B_1\sigma)$.

Let a_0, a_1, a_2, b_0, b_1, b_2, u, y_p, y_q be elemenents in $GF(3^m)$. The values of A_0, A_1, B_0, and B_1 in the ηT pairing algorithm are indicated in Table 1

A_0	$a_0 + a_1\rho + a_2\rho^2$
A_1	$b_0 + b_1\rho + b_2\rho^2$
B_0	$-u^2 - u\rho - \rho^2$
B_1	$y_p y_q$

Table 1: $A0, A1, B0, B1$

The products in $GF(3^{6m})$, for instance $C_0 + C_1\sigma + C_1\sigma^2$, are obtained by multiplications in $GF(3^{3m})$ using the Karatsuba method. The products in $GF(3^{3m})$ are obtained by $GF(3^m)$ multiplications also using the Karatsuba method. The number of multiplications in $GF(3^m)$ in $GF(3^{6m})$ depends on the multiplication method. This paragraph describes the schedule of multiplication and the number of multiplications in the ηT pairing.

- **Karatsuba Multiplication 1**
  In the Karatsuba method described in [10](Karatsuba multiplication 1), multiplication in $GF(3^{6m})$ consists of A_0B_0, $(A_0 + A_1)(B_0 + B_1)$, A_1B_1 multiplications in $GF(3^{3m})$ as shown in Table 4. Let the product in $GF(3^{3m})$ be $c_0+c_1\rho+c_2\rho^2+c_3\rho^3+c_4\rho^4, c_0, c_1, c_2, c_3, c_4 \in GF(3^m)$. c_0, c_1, c_2, c_3, c_4 as indicated in Table 5. Therefore, the $GF(3^m)$ multiplication in $GF(3^{6m})$ is calculated 17 times in the Karatsuba method 1.

C_0	A_0B_0
C_1	$(A_0 + A_1)(B_0 + B_1) - C_0 - C_2$
C_2	A_1B_1

Table 2: Product of Multiplication by Karatsuba

A_0B_0
$c_0 = -(a_0u)u$
$c_1 = -(a_0u)u - (a_1u)u + a_1u$
$c_2 = -(a_2u)u - a_1u - a_0$
$c_3 = -a_2u - a_1$
$c_4 = -a_2$
$(A_0 + A_1)(B_0 + B_1)$
$c_0 = (a_0 + b_0)(-u^2 + y_py_q)$
$c_1 = (a_1 + b_1)(-u^2 + y_py_q) - a_0u - b_0u$
$c_2 = -a_1u - b_1u - (a_2 + b_2)(-u^2 + y_py_q) - a_0 - b_0$
$c_3 = -a_2u - b_2u - a_1 - b_1$
$c_4 = -a_2 - b_2$
A_1B_1
$c_0 = b_0y_py_q$
$c_1 = b_1y_py_q$
$c_2 = b_2y_py_q$

Table 3: Multiplications by Karatsuba 1

- **Karatsuba Multiplication 2**
  In the general Karatsuba method (Karatsuba multiplication 2), multiplications in $GF(3^{6m})$ consist of $A_0B_0, (A_0 + A_1)(B_0 + B_1), A_1B_1$ multiplications in $GF(3^{3m})$ as shown as Table 6. Let the product in $GF(3^{3m})$ be $c_0 + c_1\rho + c_2\rho^2 + c_3\rho^3 + c_4\rho^4, c_0, c_1, c_2, c_3, c_4 \in GF(3^m)$. c_0, c_1, c_2, c_3, c_4 as indicated in Table 6. Therefore, the $GF(3^m)$ multiplication in $GF(3^{6m})$ is calculated 15 times in the Karatsuba method 2

3 Reuse of Intermediate Products

In this section we describe our key idea and the proposed algorithm.

Key Idea

The Shift-Addition method using the windowing algorithm[9], which is appropriate for software implementation of $GF(3^m)$ multiplication, is composed of precomputation and multiplication of polynomials.

Let k be window size. In precomputation, intermediate products are obtained by multiplying all the polynomials of degree of at most $k-1$ by the multiplier

A_0B_0
$c_0 = -a_0u^2$
$c_1 = (a_0 + a_1)(-u^2 - u) - c_0 + a_1u$
$c_2 = (a_0 + a_1 + a_2)(-u^2 - u - 1) - c_0 - c_1 - c_3 - c_4$
$c_3 = (a_1 + a_2)(-u - 1) + a_1u + a_2$
$c_4 = -a_2$
$(A_0 + A_1)(B_0 + B_1)$
$c_0 = (a_0 + b_0)(-u^2 + y_py_q)$
$c_1 = (a_0 + b_0 + a_1 + b_1)(-u^2 + y_py_q - u) - c_0 + (a_1 + b_1)u$
$c_2 = (a_0 + b_0 + a_1 + b_1 + a_2 + b_2)(-u^2 + y_py_q - u - 1)$
$-c_0 - c_1 - c_3 - c_4$
$c_3 = (a_1 + b_1 + a_2 + b_2)(-u - 1) + (a_1 + b_1)u + a_2 + b_2$
$c_4 = -a_2 - b_2$
A_1B_1
$c_0 = b_0y_py_q$
$c_1 = b_1y_py_q$
$c_2 = b_2y_py_q$

Table 4: Multiplications by Karatsuba 2

and storing them in a table. In multiplication of polynomials, the algorithm scans k coefficients at a time and accesses intermediate products by table lookup. The intermediate products, accessed by table lookup, are shifted and summed up in accumulator registers. The number of scanning and summing up operations is $\lceil m/k \rceil$.

Past research concentrated on reducing the number of multiplications as shown in Section 2 and the speed-up of multiplication such as the Shift-Addition multiplication approach.

The proposed method reduces the number of precomputations in $GF(3^m)$ multiplications for ηT pairing by reusing intermediate products. Intermediate products accessed for the multiplication of polynomials are obtained via the multiplier. In subsequent multiplication, if the multiplier is the same, the intermediate products can be reused as shown in Fig. 1(the reduction step is omitted). For instance, the intermediate products are obtained for multiplier u when u is multiplied by a multiplicand. In the following multiplication, which uses multiplier u, the intermediate products obtained from the previous multiplication are reused.

Multiplication Scheduling by Reusing Intermediate Products

As shown in Section 2 , the number of $GF(3^m)$ multiplications in $GF(3^{6m})$ depends on the multiplication method. Furthermore, the number of precomputations and multiplications of polynomials differs. Next, we determine the number of precomputations and multiplications of polynomials in $GF(3^{6m})$ in the 2 Karatsuba methods.

- **Karatsuba Multiplication 1**

  Table 8 indicates the multiplication scheduling that enables the reuse of intermediate products.

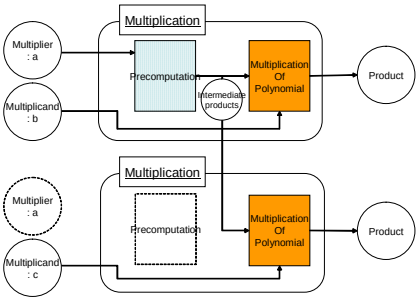

Figure 1: Reuse of Intermediate Products

u', y'_q,$(-t6 + t5)'$ and $t5'$ denote intermediate products. In this scheduling, precomputation is done 4 times and the multiplication of polynomials is done 17 times. The number of times the intermediate products can be reused is 13. The number of precomputations is reduced to 4 from 17 by the proposed algorithm.

	multiplication
1	$t0 = -a_0u$
2	$t1 = -a1u'$ (' indicates intermediate products)
3	$t2 = t1u'$
4	$t3 = -a_2u'$
5	$t3u'$
6	$t0u'$
7	$t5 = y_py_q$
8	b_0u'
9	b_1u'
10	$-b_2u'$
11	$t6 = uu'$
12	$(a_0 + b_0)(-t6 + t5)$
13	$(a_1 + b_1)(-t6 + t5)'$
14	$(a_2 + b_2)(-t6 + t5)'$
15	b_0t5
16	b_1t5'
17	b_2t5'

Table 5: Multiplication Schedule of Karatsuba method

- **Karatsuba Multiplication 2**

  Table 9 shows the multiplication scheduling that enables the reuse of intermediate products. u', $t0'$,$(-t0-u)'$,y'_q,y'_q,$(-t0+t2)'$ and $(-t0+t2-u)'$ denote intermediate products. In this scheduling, precomputation is done 7 times and the multiplication of polynomials is done 15 times. The number of times the intermediate products can be reused is 8. The number of precomputations is reduced to 7 from 15 by the proposed algorithm.

	multiplication
1	$t0 = uu$
2	$a_0 t0$
3	$t1 = a_1 u'$ (' indicates intermediate products)
4	$(a_0 + a_1)(-t0 - u)$
5	$(a_1 + a_2)u'$
6	$(a_0 + a_1 + a_2)(-t0 - u)'$
7	$t2 = y_p y_q$
8	$(a_0 + b_0)(-t0 + t2)$
9	$(a_1 + b_1)u$
10	$(a_0 + b_0 + a_1 + b_1)(-t0 + t2 - u)$
11	$(a_1 + b_1 + a_2 + b_2)u'$
12	$(a_0 + b_0 + a_1 + b_1 + a_2 + b_2)(-t0 + t2 - u)'$
13	$b_0 t2$
14	$b_1 t2'$
15	$b_2 t2'$

Table 6: Multiplication Schedule of Karatsuba method

4 Efficient Implementation

In this section, we describe an efficient computation implementation for the polynomial multiplication in $GF(3^m)$. Let k be window size. Let m be the degree of polynomials. Let w be the word size of the processor. The precomputation is composed of $3^k - 2 - (k-1)$ times addition and $k-1$ times shift operations. The multiplication of polynomials is composed of $\lfloor m/k \rfloor$ times addition and shift operations. If the maximum degree of polynomials is below the word size of the processor, addition is composed of 7 instructions[11] and a shift operation is composed of 2 shift instructions. A good compromise between the speedup and the number of precomputations is to set $k = 3$. When using $k = 3$, the computing speed of the polynomial multiplication is slow compared with the precomputation speed. Thus speed-up of polynomial multiplication is helpful for reducing more efficient ηT pairing computation. We show an efficient method of computating the multiplication of polynomials in this section.

Prior Implementation Approach

The coefficients of the polynomials are represented by hi-bit registers and low-bit registers[11]. The bits that express the coefficients are arranged in order of either LSB or MSB.Owning to this, it is necessary to pass the intermediate product between two or more registers(multi-bit shift operation) in the multiplication of polynomials. To pass the intermediate product between registers, a rotate shift instructions or double shift instructions are commonly used. Rotate shift instructions are not available here because they cannot pass two bits or more at a time. To pass values between multiple registers, double shift instructions are used for the above multi-bit shift operation, however the operation cost of a double shift instruction is higher than that of a shift instruction[12].

Modified Shift-Addition Multiplication

We propose a new polynomial multiplication strategy that does not use the above multi-bit shift operation. Let $I(x)$ be an intermediate product looked up in a table. Let m' be the degree of the intermediate product. Let w be the word size of the target processor. The intermediate product $I(x)$ is divided into $I'(x)_i x^{i(\lceil \frac{m'}{\lceil m'/w \rceil} \rceil}$ and loaded into the registers as shown in Fig. 2 where $I(x) = \sum_{i=0}^{\lceil m'/w \rceil - 1} I'(x)_i x^{i(\lceil \frac{m'}{\lceil m'/w \rceil} \rceil}$

The intermediate product, accessed by table lookup, is shifted and summed up in accumulator registers. Modified Shift-Addition multiplication does not pass the intermediate product between two or more registers in the shift operation. The value loaded in each register is shifted respectively(respective shift operation). This operation makes two or more values of the same degree appear in the accumulator registers as illustrated in Fig. 3. To obtain the result, adding the coefficients of the same degree is required.

Multiplication of polynomials cannot be achieved by just shift instructions; it requires double shift instructions because the degree of multiplicand polynomial is greater than $m - \lceil \frac{m'}{\lceil m'/w \rceil} \rceil$. To compose the polynomial multiplication of more shift instructions, the multiplicand is divided into partial multiplicands of degree at most $(m - (\lceil \frac{m'}{\lceil m'/w \rceil} \rceil) + k - 1$. The partial products obtained by multiplying the partial multiplicands by intermediate products are summed up and placed in accumulator registers.

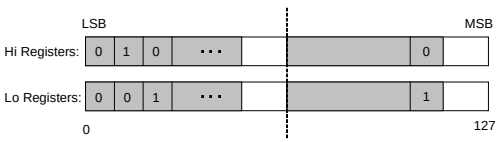

Figure 2: Modified Shift-Addition Multiplication 1

Implementation on a 64-bit Processor

We implemented modified Shift-Addition multiplication in $GF(3^{97}), k = 3$ on 64-bit platform in assembly code. We chose Athlon FX 57 which implements the Intel instruction set. The FX 57 contains 2MB of second cache and 16 64-bit registers. The FX57 is clocked at 2.8MHz. Multiplication of polynomials is composed of 3 steps as follows:

- **Obtaining Intermediate Products**
  When $k = 3$, intermediate products are obtained by multiplying all trinomials by the multiplier

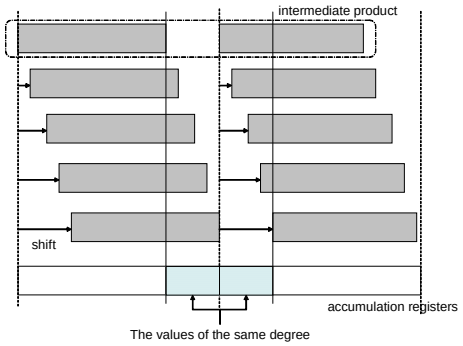

Figure 3: Modified Shift-Addition Multiplication 2

and reducing with irreducible trinomial $x^{97}+x^{12}+2$. Though usually the reduction step is done after the multiplication of polynomials step, by reducing intermediate products, the number of $m - \lceil \frac{m'}{\lceil m'/w \rceil} \rceil$ can be enlarged to $m - \lceil \frac{m}{\lceil m/w \rceil} \rceil$. This reduces the number of partial products.

By this reduction, the maximum degree of intermediate products is 96. Let $P_i, 0 < i < 27$ be intermediate products. P_i is divided into $P_{iLSB} + P_{iMSB}x^{49}$. P_{iLSB} and P_{iMSB} are stored in 64-bit registers. Registers holding P_{iLSB} have 30 empty bits and registers holding P_{iMSB} have 32 empty bits.

- **Multiplication of Polynomials**
  In the multiplication $C(x) = A(x)B(x)$, the multiplicand $A(x)$ is divided into 5 partial multiplicands. To obtain the product, the partial products obtained by multiplying the partial multiplicands by the multiplier are summed up as $C(x) = \sum_{i=0}^{5} C_i$ as shown below.

$$\begin{aligned} C_0(x) &= (a_0 + a_1x + ... + a_{17}x^{17}) \\ C_1(x) &= (a_{18} + a_{19}x + ... + a_{35}x^{17})x^{18} \times B(x) \\ C_2(x) &= (a_{36} + a_{37}x + ... + a_{53}x^{17})x^{36} \times B(x) \\ C_3(x) &= (a_{54} + a_{55}x + ... + a_{71}x^{17})x^{54} \times B(x) \\ C_4(x) &= (a_{72} + a_{73}x + ... + a_{89}x^{17})x^{72} \times B(x) \\ C_5(x) &= (a_{90} + a_{91}x + ... + a_{96}x^{6})x^{90} \times B(x) \end{aligned} \quad (1)$$

- **Computation of Partial Products**
  In the computation of $C_i(x)$, untile empty bits of registers, where P_{iLSB} and P_{iMSB} are stored, are disappeared, the value of each register is shifted and added in $GF(3^{97})$ repeatedly. After 5 times shifting and adding, 2 values of the same degree from x^{49} to x^{63} appear in the accumulator registers. Next, P_{iLSB} is divided into P_{iLSB1} and $P_{iLSB2}x^{49}$ and $P_{iLSB2}$and $P_{iMSB}$are added. The above-mentioned step calculates the partial product. The calculation of the partial product was implemented by 5 multi-bit shift operations and 10 additive operations. Our method replaces 5 multi-bit shift operations by 5 shift operations and 1 addition.

5 Results and Comparison

In this section, we show the time taken by the precomputation and multiplication of polynomials using the modified Shift-Addition multiplication algorithm on a 64-bit processor. Next, by multiplying the time taken for multiplication in $GF(3^{97})$ by the number of multiplications in $GF(3^{6m})$, we obtain the time taken for multiplications in $GF(3^{6m})$ in the ηT pairing.

Evaluation of modified Shift-Addition multiplication

Table 10 shows the evaluation environment. Table 11 shows average modified Shift-Addition multiplication execution times. Average times are estimated by 100000000 times execution. Shift-Addition multiplication time was measured on an AMD Opteron Processor 275 clocked at 2.2GHz. To allow a valid comparison, we multiply $0.5009\mu s$ by 2.2/2.8.

Table 7: Measurement environment

CPU	Athlon FX 57 2.8GHz
memory	2GBytes
OS	SUSE Linux 10.1
Language	C/assembly
Compiler	gcc 4.1.0
Compiling option	-O2

Time Taken by Multiplications in $GF(3^{6\cdot 97})$ in ηT Pairing

The computation timing of $GF(3^m)$ multiplications in $GF(3^{6m})$ of ηT pairing is shown in Table 12. This result shows that our algorithm, which reuses the intermediate products, yielded a 40% performance increase for multiplications of $GF(3^m)$ in $GF(3^{6m})$ in the ηT pairing on an AMD 64-bit processor. Moreover, the computation time of Karatsuba 1 is shorter than that of Karatsuba 2 though the number of multiplications is slightly fewer in Karatsuba 2. Namely, the reduction in the number of the polynomial multiplications is more important than the reduction in the multiplication time; a result that goes against previous research on speeding up the ηT pairing.

6 Conclusion and Future Research Topics

We have presented a novel multiplication algorithm $GF(3^{6m})$ that signifcantly reduces the number of precomputations in ηT pairing and an efficient implementation approach that is suitable for ηT pairing. These

Table 8: Comparison with prioir methods(μs)

	Precomputation	Multiplication of polynomials	Multiplication
Modified Shift-Addition	0.13485	0.15509	0.28994
Shift-Addition	0.395		

Table 9: Comparison with prior methods(μs)

Multiplication Schedule		Precomputation	Multiplication of polynomial	in $GF(3^{6\cdot 97})$
Karatsuba 1	Proposed Method	0.5394	2.63653	3.17593
	Prior Method	0.28994		4.92898
Karatsuba 2	Proposed Mehotd	0.94395	2.32635	3.2703
	Prior Method	0.28994		4.3491

algorithms led to a 40% performance increase for multiplications of $GF(3^m)$ in $GF(3^{6m})$ in ηT on an AMD 64-bit processor. In this article, to obtain the time taken by multiplications in $GF(3^{6m})$ in the ηT pairing, the time taken by multiplication in $GF(3^{97})$ was multiplied by the number of multiplications in $GF(3^{6m})$

In the future, the computation time of the ηT pairing over $GF(3^{97})$, $GF(3^{163})$, $GF(3^{197})$, $GF(3^{239})$, $GF(3^{313})$ will be measured. We plan to verify the effectiveness of the proposed methods on 32-bit, 16-bit and 8-bit CPUs. There are many multiplication schedules other than the Karatsuba method, for instance FFT[16]. We also plan to implement multiplication of $GF(3^{6m})$ on other schedules. The adjustment area and the effect of the proposed methods will be clarified.

References

[1] D. Boneh and M.K. Franklin, "Identity-Based Encryption from the Weil Pairing", Proceedings of CRYPTO 2001 on Advances in cryptology, 2001

[2] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing", Proceedings of ASIACRYPT 2001 on Advances in cryptology, 2001

[3] H. COHEN and G. Frey, "Handbook of Elliptic and Hyperelliptic Curve Cryptography", Discrete Mathematics and Its Applications. Chapman and Hall/CRC, 2006, pp176–177

[4] T. Kerins, W. P. Marnane, E. M. Popovici, and P.S.L.M. Barreto, "Efficient hardware for the Tate pairing calculation in characteristic three", Cryptographic Hardware and Embedded Systems, CHES 2005. Volume 3659 of Lecture Notes in Computer Science,pp412–426, Springer-Verlag

[5] T. Izu and T. Takagi, Efficient Computations of the Tate Pairing for the Large MOV Degrees, Conference on Information Security and Cryptology, ICISC 2002

[6] I.M Duursma and H.-S Lee, Tate Pairing implementation for hyperelliptic curves, Advances in Cryptology ASIACRYPT 2003

[7] P. S. L. M. Barreto and M. Naehrig, Pairing-friendly elliptic curves of prime order, Selected Areas in Cryptography?SAC 2005

[8] M. Shirase, T. Takagi, and E. Okamoto, "Some Efficient Algorithms for the Final Exponentiation of ηT Pairing", 3rd Information Security Practice and Experience Conference, ISPEC 2007, LNCS 4464, pp.254-268, 2007, Springer-Verlag.

[9] H. COHEN and G. Frey, "Handbook of Elliptic and Hyperelliptic Curve Cryptography",ser. Discrete Mathematics and Its Applications. Chapman and Hall/CRC, 2006, pp216–217

[10] J. Beuchat, M. Shirase, T. Takagi, and E. Okamoto, "An Algorithm for the ηT Pairing Calculation in Characteristic Three and its Hardware Implementation", the 18th IEEE International Symposium on Computer Arithmetic.

[11] K.Harrison, D.Page, and N.P. Smart, "Software Implementation of Finite Fields Of Characteristic Three, For User In Pairing-Based Cryptosystems", LMS Journal Computation and Mathematics, Vol 5, pp 181-193, 2002

[12] AMD, Software Optimization Guide for AMD64 Processor, 2005

[13] AMD, AMD Athlon 64 FX Processor, http://www.amd.com/

[14] Y. Karahara, M. Shirase, T. Takagi, and E. Okamoto, "Efficient Software Implimentation of ηT Pairing" , Symposium on Cryptography and Information Security 2007, 2E3-2, 2007

[15] R. Rivest, A. Shamir, and L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM 1978

[16] E. Gorla, C. Puttmann, and J, Shokrolliahi "Efficient formulas for efficient multiplication in F_3^{6m} " The 22nd Annual ACM Symposium on Applied Computing, 2007