Efficient Implementation of Tate Pairing on a Mobile Phone using Java

Yuto Kawahara[1], Tsuyoshi Takagi[1], and Eiji Okamoto[2]

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

Abstract. Pairing-based cryptosystems (PBC) have been attracted by researchers in cryptography. Some implementations show that PBC are relatively slower than the standard public key cryptosystems. We present an efficient implementation for computing Tate pairing on a mobile phone using Java. We implemented the η_T pairing (a recent efficient variation of Duursma-Lee algorithm) over some finite fields of characteristic 3 with extension degree $m = \{97, 167, 193, 239\}$. Our optimized implementation for $m = 97$ achieved about 0.5 seconds for computing Tate pairing over FOMA SH901iS, NTT DoCoMo. Then our implementation of Tate pairing is compared in the same platform with other Java program of the standard cryptosystems, i.e., RSA cryptosystem and elliptic curve cryptosystem (ECC). The computation speed of Tate pairing is comparable to that of RSA or ECC on the same mobile device.

Keywords: Tate pairing, Java, mobile phone, efficient implementation.

1 Introduction

Pairing-based cryptosystems (PBC) can provide us several novel cryptographic applications, e.g., ID-based cryptosystems [5], short digital signatures [7], broadcast encryption [6], etc. Some of them have not been achieved using the conventional public key cryptosystems. Therefore PBC have been attracted by researchers in cryptography. PBC use the Tate pairing on elliptic curves over finite fields. The standard algorithm for computing the Tate pairing is Miller's algorithm. Miller's algorithm is about 5 times slower than 1024-bit RSA and 160-bit elliptic curve cryptosystem (ECC) [2]. It is an important research topic to find more efficient algorithms for computing Tate pairing.

Recently, Duursma and Lee introduced an efficient implementation of Miller's algorithm specified for supersingular curves over finite fields $\mathbb{F}_{3^m}$ [8]. The order of the supersingular curves has the very low Hamming weight (i.e. 3) and this algorithm can be implemented in a closed form only using the arithmetic of the underlying finite field. Kwon then presented an efficient variation of Duursma-Lee algorithm without computing cube roots [15]. This algorithm over $\mathbb{F}_{3^{97}}$ has been implemented in several milliseconds on FPGA [14] or Pentium using C language [18, 1]. Moreover, Barreto et al. proposed the η_T pairing algorithm that is about twice faster than Duursma-Lee algorithm [1]. The number of the loops in η_T pairing algorithm becomes $(m + 1)/2$ using an endomorphism map, which is about the half of m used in Duursma-Lee algorithm. η_T pairing over $\mathbb{F}_{2^m}$ can be implemented under one second on a smart card [16].

Java 2 Platform, Micro Edition (J2ME) is a secure and flexible Java platform designed for embedding devices such as mobile phones [12]. Some applications of pairing based cryptosystems are suitable for the environments using ubiquitous devices. Java provides several security components of the standard public-key cryptosystem (RSA, ECC) and other cryptographic functions [13]. Tillich and Großschadl presented some implementation of the standard public-key cryptosystem over mobile phones [19]. However, no implementation of the pairing-based cryptosystems using Java over mobile phones has been reported.

In this paper, the feasibility of Tate pairing on mobile phones using Java is examined. We implement η_T pairing algorithm over $\mathbb{F}_{3^m}$ for several extension degrees $m = \{97, 167, 193, 239\}$, and especially optimized it for extension degrees $m = 97$. Because there is no JCE component for computing $\mathbb{F}_{3^m}$, we have to implement the arithmetic of $\mathbb{F}_{3^m}$ from scratch. We design that the number of Java components is preferable to be as small as possible, and avoid a large overhead in the computation speed when many Java components are called. The speed of calculations in $\mathbb{F}_{3^m}$ strongly depends on the choice of the representation of elements, so that we deploy the bit representation suitble for basic logic operations in Java and the irreducible trinomial basis $x^m + x^k + 2$ with smallest degree k. In this implementation, our program achieves about 0.5 seconds on FOMA SH901iS, NTT DoCoMo. Moreover, our implementation of Tate pairing is compared with the standard public-key cryptosystems (1024-bit RSA and 160-bit ECC) using the Java components provided by Bouncy Castle [21]. Then the speed of our implementation is comparable to that of the standard public-key cryptosystems with the same security parameters.

2 Arithmetic in Finite Fields with Characteristic 3

In this section, we describe the arithmetic of finite fields with characteristic 3 of degree m, where m is positive integer. We denote by $\mathbb{F}_{3^m}$ the finite fields.

2.1 Bit Representation of $\mathbb{F}_{3^m}$

Let $\mathbb{F}_3 = \{0, 1, 2\}$ be the finite field with characteristic 3. The element a in $\mathbb{F}_3$ is encoded by two bits such as $a = (a_{hi}, a_{lo})$ for $a_{hi}, a_{lo} \in \{0, 1\}$. In the implementation we choose $0 = (0, 0), 1 = (0, 1)$, and $2 = (1, 0)$. The negative $-a$ for $a \in \mathbb{F}_3$ is replaced by $2a$.

Every elements in $\mathbb{F}_{3^m}$ are represented $\mathbb{F}_3[x]/f(x)$, where $f(x)$ is an irreducible polynomial of degree m in $\mathbb{F}_3[x]$. Let $A(x)$ be an element in $\mathbb{F}_{3^m}$. $A(x)$ can be represented as the polynomial of degree at most $m-1$, namely $\sum_{i=0}^{m-1} a_i x^i$ $(a_i \in \mathbb{F}_3)$ [4]. For our implementation, each coefficient a_i is represented as follows:

$$A_{hi} = ((a_{m-1})_{hi}, (a_{m-2})_{hi}, \cdots, (a_0)_{hi}), \; A_{lo} = ((a_{m-1})_{lo}, (a_{m-2})_{lo}, \cdots, (a_0)_{lo})$$

Here we denote $A(x)$ by (A_{hi}, A_{lo}). Using this representation, we need an array with size of $N = (m/W) + 1$ for storing an element in $\mathbb{F}_{3^m}$, where W is the word size of the target processor. Note that a negative element $-A(x)$ is replaced $2A(x)$ by changing A_{hi} and A_{lo}.

2.2 Addition and Multiplication in $\mathbb{F}_{3^m}$

Let $A(x) = (A_{hi}, A_{lo})$ and $B(x) = (B_{hi}, B_{lo})$ be the elements of $\mathbb{F}_{3^m}$. Addition $C(x) = (C_{hi}, C_{lo}) = A(x) + B(x)$ is performed by the basic logic operators (AND(&), OR(|) and XOR($\wedge$)) as follows:

$$T = (A_{hi} \mid A_{lo}) \& (B_{hi} \mid B_{lo}),\ C_{hi} = T \wedge (A_{hi} \mid B_{hi}),\ C_{lo} = T \wedge (A_{lo} \mid B_{lo}).$$

Subtraction $A(x) - B(x)$ can be computed as $A(x) + B'(x)$ for $B'(x) = 2B(X)$.

Multiplication in $\mathbb{F}_{3^m}$ consists of the polynomial multiplication step and the reduction step. The polynomial multiplication step computes $C'(x) = \sum_{i=0}^{2m-2} c'_i x^i = A(x) \cdot B(x)$ for given $A(x), B(x)$ in $\mathbb{F}_{3^m}$. The shift-addition multiplication method is the simplest algorithm, and other methods (e.g. the comb method) require additional memory comparing with the shift-addition multiplication method [10]. Therefore it is suitable for the implementation on memory constraint device such as mobile phones. Let $A(x) = \sum_{i=0}^{m-1} a_i x^i$ and $B(x) = \sum_{i=0}^{m-1} b_i x^i$ be the elements of $\mathbb{F}_{3^m}$. The polynomial multiplication step $C'(x) = \sum_{i=0}^{2m-2} c'_i x^i = A(x) \cdot B(x)$ is performed by the following algorithm.

Algorithm 1 Shift-Addition Multiplication in $\mathbb{F}_{3^m}$

```
Input: $A(x) = \sum_{i=0}^{m-1} a_i x^i, B(x) = \sum_{i=0}^{m-1} b_i x^i \in \mathbb{F}_{3^m}$  $(a_i, b_i \in \mathbb{F}_3)$
Output: $C'(x) = A(x) \cdot B(x)$
1: $C'(x) \leftarrow 0$
2: for $i \leftarrow 0$ to $m-1$ do
3:     $C'(x) \leftarrow C'(x) + b_i A(x) \cdot x^i$
4: end for
```

Next, the reduction step is computed based on the irreducible polynomial of $\mathbb{F}_{3^m}$. In order to accelerate the reduction step, we deploy the irreducible trinomial $f(x) = x^m + x^k + 2$, where $m > k > 1$. The reduction algorithm with irreducible trinomial $f(x)$ is described as follows:

Algorithm 2 Reduction with Trinomial Basis

```
Input: $C'(x) = \mathrm{GF}(3)[x]$ of degree $n$ $(n > m-1)$, $f(x) = x^m + x^k + 2$
Output: $C'(x) = C'(x) \bmod f(x)$
1: for $i \leftarrow n$ downto $m$ do
2:     $c'_{i-k} \leftarrow c'_{i-k} - c'_i$
3:     $c'_{i-m} \leftarrow c'_{i-m} - 2c'_i$
4:     $c'_i \leftarrow 0$
5: end for
```

2.3 Other Operations in $\mathbb{F}_{3^m}$

We describe other operations in $\mathbb{F}_{3^m}$, namely cube, inversion and cube root.

Cube: For a given polynomial $A(x) \in \mathbb{F}_3[x]$, the cube of $A(x)$ is calculated by $(A(x))^3 = \sum_{i=0}^{m-1} a_i x^{3i}$. Therefore the multiplication step is computed virtually for free. We present two methods for the reduction step of the polynomial $\sum_{i=0}^{m-1} a_i x^{3i}$. One performs the standard reduction algorithm using the irreducible trinomial $x^m = 2x^k + 1$. Another one utilizes a reduction table, which directly finds the reduced polynomial for a given $A(x) \in \mathbb{F}_3[x]$. The speed of the second reduction method can be optimized for fixed degree m. However the reduction table should be prepared depending on the extension degree m, and the first reduction method is suitable for general degree m.

Inversion: Inversion is performed using the extended Euclidean algorithm for the polynomials over $\mathbb{F}_3[x]$. We developed a ternary version of the extended Euclidean algorithm over binary polynomial $\mathbb{F}_2[x]$ [10]. The details of our algorithm is described in appendix A. Another algorithm for computing an inversion is the ternary gcd [11]. In our experiment, the ternary extended Euclidean algorithm is faster than the ternary gcd.

Cube Root: Cube root is efficiently implemented by the algorithm proposed by Barreto et al. [3]. One cube root can be computed with the speed of at most two multiplications in $\mathbb{F}_{3^m}$.

2.4 Arithmetic in Extension Field $\mathbb{F}_{3^{6m}}$

Extension field $\mathbb{F}_{3^{6m}}$ is represented $\mathbb{F}_{3^{3m}}[\sigma]/h(\sigma)$ and $\mathbb{F}_{3^m}[\rho]/g(\rho)$, where $h(\sigma)$ and $g(\rho)$ are irreducible polynomials $h(\sigma) = \sigma^2 + 1$ over $\mathbb{F}_{3^{3m}}$ and $g(\rho) = \rho^3 - \rho - 1$ over $\mathbb{F}_{3^m}$. Let $A(\sigma), B(\sigma)$ be elements in $\mathbb{F}_{3^{6m}}$. We denote by $A(\sigma) = a_1\sigma + a_0$ the element in $\mathbb{F}_{3^{6m}}$, where a_0, a_1 are elements in $\mathbb{F}_{3^{3m}}$. The operations in $\mathbb{F}_{3^{6m}}$ are implemented as follows:

- **Addition:** $A(\sigma) + B(\sigma) = (a_1 + b_1)\sigma + (a_0 + b_0)$.
- **Multiplication :** $A(\sigma)B(\sigma) = (t_{00} - t_{11})\sigma + (t_{01} - t_{00} - t_{11})$, where $t_{00} = a_0 b_0,\ t_{11} = a_1 b_1,\ t_{01} = (a_0 + a_1)(b_0 + b_1)$. Multiplication in $\mathbb{F}_{3^{6m}}$ can be obtained 18 multiplications and 57 additions in $\mathbb{F}_{3^m}$.
- **Cube:** $A(\sigma)^3 = -a_1^3\sigma + a_0^3$. Cube in $\mathbb{F}_{3^{6m}}$ is computed by 6 cubes and 4 addition in $\mathbb{F}_{3^m}$.
- **Inversion:** $A(\sigma)^{-1} = t^{-1}(a_0 - a_1\sigma)$,where $t = (a_1^2 + a_0^2)$. Inversion in $\mathbb{F}_{3^{6m}}$ uses 1 inversion, 21 multiplications and 45 additions in $\mathbb{F}_{3^m}$.

3 Implementation of Tate pairing

In this paper, we implement the Tate pairing on the following supersingular elliptic curve over $\mathbb{F}_{3^m}$,

$$E(\mathbb{F}_{3^m}) = \{(x, y) \in \mathbb{F}_{3^m} \mid y^2 = x^3 - x + 1\}.$$

A point over curve $E(\mathbb{F}_{3^m})$ is represented by (x, y), where x and y are elements in $\mathbb{F}_{3^m}$. All points on $E(\mathbb{F}_{3^m})$ with the point of infinity forms a group structure. The group order of curve $E(\mathbb{F}_{3^m})$ is $\#E = 3^m + 3^{(m+1)/2} + 1$. The pairing based cryptosystems require the arithmetic of curve $E(\mathbb{F}_{3^m})$ such as point addition, point double, point tripling, and scalar multiplication [11].

Let $P = (x_p, y_p), Q = (x_q, y_q)$ be input points over elliptic curve $E(\mathbb{F}_{3^m})$. We describe some formulae on $E(\mathbb{F}_{3^m})$ in the following.

- **Point Addition:** Point addition $(x_r, y_r) = P + Q$, $(x_p \neq x_q)$ is implemented by the following algorithm $x_r = \lambda^2 - (x_p + x_q)$, $y_r = (y_p + y_q) - \lambda^3$, where $\lambda = (y_q - y_p)/(x_q - x_p)$. Point addition needs 2 multiplications, 1 inversion, 1 cube, and 6 additions in $\mathbb{F}_{3^m}$.
- **Point Double:** Point double $(x_r, y_r) = P + P$ is computed by $x_r = \lambda^2 + x_p$, $y_r = -(y_p + \lambda^3)$, where $\lambda = 1/y_p$. Point double needs 1 multiplication, 1 inversion, 1 cube and 2 additions in $\mathbb{F}_{3^m}$.
- **Point Tripling:** Point tripling $(x_r, y_r) = 3P = P + P + P$ is computed by $x_r = ((x_p)^3)^3 - 1$, $y_r = -((y_p)^3)^3$. Point tripling requires only 4 cubes in $\mathbb{F}_{3^m}$ and it is very efficiently computed.
- **Scalar Multiplication:** Scalar multiplication is defined by dP, where P is a point on $E(\mathbb{F}_{3^m})$ and d is a integer. This algorithm is calculated by triple-and-addition algorithm [2], and can be obtain $\lfloor log_3 d \rfloor$ point triplings, about $\frac{2}{3}\lfloor log_3 d \rfloor$ point additions and 1 point double.

Let l be a large prime number that satisfies $l | \#E$ and $l | (3^{6m} - 1)$. Denote by $E(\mathbb{F}_{3^m})[l]$ the subgroup of $E(\mathbb{F}_{3^m})$ with order l. Let $\phi(x, y) = (-x + \rho, y\sigma)$ be the distortion map, which maps a point $Q = (x, y)$ on $E(\mathbb{F}_{3^m})[l]$ to the point $\phi(Q)$ in $E(\mathbb{F}_{3^{6m}})[l]$ the elliptic curve defined over the extension field $\mathbb{F}_{3^{6m}}$. The pairing $e(P, Q)$ is a bilinear map

$$\begin{array}{rcl} e : E(\mathbb{F}_{3^m})[l] \times E(\mathbb{F}_{3^{6m}})[l] & \to & \mathbb{F}^*_{3^{6m}}/(\mathbb{F}^*_{3^{6m}})^l \\ (P, \phi(Q)) & \mapsto & e(P, Q), \end{array}$$

which satisfies $e(aP, Q) = e(P, aQ) = e(P, Q)^a$ for every non-zero integer a.

3.1 Implementation of η_T Pairing

Miller's algorithm is the standard algorithm for computing Tate pairing, but it is about 5 time slower than the standard public-key cryptosystems [2]. Duursma-Lee algorithm is a simply modified version of Miller's algorithm for supersingular curves over $\mathbb{F}_{3^m}$ [8]. The order of the supersingular curve has the very low Hamming weight (i.e. $\#E = 3^m + 3^{(m+1)/2} + 1$) and this algorithm can be implemented in a closed form (any operators with the arithmetic in $\mathbb{F}_{3^m}$). Barreto et al. proposed a faster variation of Duursma-Lee algorithm called η_T pairing [1]. The number of the loops in η_T pairing algorithm is reduced to the half of that in Duursma-Lee algorithm, and η_T pairing is about twice faster than Duursma-Lee. We have the relationship $\eta_T(P, Q)^{3(3^{(m+1)/2}+1)^2} = e(P, Q)^{-3^{(m+3)/2}}$ for given two points $P, Q \in E(\mathbb{F}_{3^m})[l]$. We describe the algorithm of η_T pairing in the following.

Algorithm 3 η_T pairing on $E(\mathbb{F}_{3^m}) : y^2 = x^3 - x + 1,\ m = \pm 1 \bmod 12$

```
Input: $P = (x_p, y_p), Q = (x_q, y_q) \in E(\mathbb{F}_{3^m})$
Output: $\eta_T(P, Q) \in \mathbb{F}_{3^{6m}}$
 1: $y_p \leftarrow -y_p$  (in $\mathbb{F}_{3^m}$)
 2: $f \leftarrow y_q\sigma - y_p(x_p + x_q + 1) + y_p\rho$  (in $\mathbb{F}_{3^{6m}}$)
 3: for $i \leftarrow 0$ to $(m-1)/2$ do
 4:     $u \leftarrow x_p + x_q + 1$  (in $\mathbb{F}_{3^m}$)
 5:     $g \leftarrow y_p y_q \sigma - u^2 - u\rho - \rho^2$  (in $\mathbb{F}_{3^{6m}}$)
 6:     $f \leftarrow fg$  (in $\mathbb{F}_{3^{6m}}$)
 7:     $x_p \leftarrow x_p^{1/3},\ y_p \leftarrow y_p^{1/3}$  (in $\mathbb{F}_{3^m}$)
 8:     $x_q \leftarrow x_q^3,\ y_q \leftarrow y_q^3$  (in $\mathbb{F}_{3^m}$)
 9: end for
10: return $f^{(3^{3m}-1)(3^m+1)(3^m-3^{(m+1)/2}+1)}$
```

The final exponential $f^{(3^{3m}-1)(3^m+1)(3^m-3^{(m+1)/2}+1)}$ can be efficiently computed [14, 1]. Indeed, we can use equation $f^{(3^{3m}-1)} = (f_0 - f_1\sigma)(f_0 + f_1\sigma)^{-1}$ for $f = f_0 + f_1\sigma$, and remaining exponent can be computed by $2m$ cubes, 2 multiplications and 1 inversion in $\mathbb{F}_{3^{6m}}$.

4 Implementation using Java

Java is a multi-platform language, which is suitable for the programming on ubiquitous devices and can develop security systems more easily comparing to some other program languages such as C. Java Micro Edition (J2ME) is one of component set of Java 2, and often uses embedded devices such as mobile phone. Java cryptography extension (JCE) is component library provide cryptography communication functions. Java language supports JCE which has the standard public key cryptosystem components, e.g. RSA cryptosystem and elliptic curve cryptosystems etc [13]. Cryptographic components are provided by some institutes, for examples, Bouncy Castle [21] and IAIK [22]. On the other hands, the pairing-based cryptosystems (PBC) accomplish novel security applications on ubiquitous environments. Therefore it is an interesting research topic to implement PBC on ubiquitous devices. However, no implementation of PBC on mobile phones has been reported.

In this paper, we try to implement Tate pairing from scratch in Java. There is a large overhead in computation speed when many components of J2ME are called. The goal of our implementation is to develop efficient computing of Tate pairing, so that the number of Java components is preferable to be as small as possible. Indeed, our program has a simply structure of Java class that contains variables and methods. We implement six Java classes, i.e. finite field parameters, finite field, extension field of extension degree 3, extension field of extension degree 6, elliptic curve point, and Tate pairing.

Another goal is to implement Tate paring by a general-purpose program that can compute by various extension degree m and irreducible trinomial $f(x)$. This is important to enhance the security by increasing the degree m in the future. The general-purpose program has the following variables of finite field $\mathbb{F}_{3^m}$ appeared in Section 2, namely the characteristic 3, the extension degree

m, the middle degree of the irreducible trinomial k, the array size N, and the word size W. Then this program can support various extension degrees with different irreducible trinomials.

On the other hand, we develop an optimized program for Tate paring with $\mathbb{F}_{3^{97}}$. The optimized program does not have the above variables, in other words does not have the finite field parameters class, but this includes directly values that correspond to these variables in algorithm. Moreover we improve multiplication and cube for the optimized program, and thus this program is computed faster.

4.1 Timing Results on a Mobile Phone

In this section, we describe timing results of our implementation using Java. We use a mobile phone FOMA SH901iS, NTT DoCoMo in order to measure timing result. We utilize degree $m = \{97, 167, 193, 239\}$, and the irreducible trinomial $f(x)$ used for each degree m is as follows: $x^{97} + x^{12} + 2,\ x^{167} + x^{96} + 2,\ x^{193} + x^{12} + 2,\ x^{239} + x^{24} + 2$, respectively. Table 2 presents the timing of the η_T pairing over $\mathbb{F}_{3^m}$ and the arithmetic in $\mathbb{F}_{3^m}$ appeared in Section 2. We show the timing of using the general-purpose program by degree $m = \{97, 167, 193, 239\}$. The optimized program for extension degree $m = 97$ is denoted by "$opt\mathbb{F}_{3^{97}}$". The timing results of $\mathbb{F}_{3^m}$ and η_T pairing are shown in the following. All timings are estimated on average by randomly chosen one thousand elements.

Table 1. Timing of Tate Pairing with Several $\mathbb{F}_{3^m}$ (msec)

Operator	$opt\mathbb{F}_{3^{97}}$	$\mathbb{F}_{3^{97}}$	$\mathbb{F}_{3^{167}}$	$\mathbb{F}_{3^{193}}$	$\mathbb{F}_{3^{239}}$
Addition	0.0173	0.0171	0.0202	0.0203	0.0198
Subtraction	0.0196	0.0193	0.0225	0.0232	0.0210
Multiplication	0.2400	0.2897	0.6651	0.8638	1.1891
Cube	0.0473	0.0886	0.1149	0.1254	0.1362
Inversion	1.5288	1.5411	3.7500	5.0203	6.6621
Cube Root	0.2982	0.3701	0.5112	0.6094	0.7941
η_T pairing	509.22	627.65	1724.93	2368.58	3557.42

Addition, subtraction and cube implemented by the general-purpose program in $\mathbb{F}_{3^m}$ are computed in almost same speed for each degree. The other operations (multiplication, inversion and cube root) become gradually slower as the degree m increases. Here η_T pairing algorithm uses many multiplications and cubes, whose number increases based on the degree m due to the number of loop. Therefore the timing of η_T pairing also become slower in terms of increasing the extension degree m. For example, the timing of η_T pairing over $\mathbb{F}_{3^{239}}$ is about 6 times slower than that over $\mathbb{F}_{3^{97}}$.

On the other hands, the timing of η_T pairing by the optimized program over $opt\mathbb{F}_{3^{97}}$ is about 1.2 time faster than that by the general-purpose program, because the optimized program is implemented using less variables, and has more efficient algorithms for multiplication and cube.

4.2 Comparison to Standard Cryptosystems

In this section, we compare our implementation of η_T pairing over $\mathbb{F}_{3^{97}}$ to the standard public key cryptosystems, i.e. 1024-bit RSA and 160-bit elliptic curve cryptosystem (ECC) over $\mathbb{F}_{2^{163}}$. Other papers show that the size of these parameters have the same security level [2].

We implement a modular exponentiation for 1024-bit RSA and a scalar multiplication for 160-bit ECC over $\mathbb{F}_{2^{163}}$ using components distributed by Bouncy Castle [21]. The modular exponentiation is (T^d mod n) for given 1024-bit integers T, d, n. We use the parameters of ECC from Certicom [20], and the scalar multiplication is dP for a given 160-bit integer d and a point P. Timing results for each cryptosystem are as follows:

Table 2. Comparison of Timing with Other Cryptosystems (msec)

Operator	FOMA SH901iS	Pentium M
η_T pairing with $\mathbb{F}_{3^{97}}$	509.22	10.15
Modular exponentiation of 1024-bit RSA	4238.40	75.07
Scalar multiplication of ECC over $\mathbb{F}_{2^{163}}$	13777.50	116.83

η_T pairing with $\mathbb{F}_{3^{97}}$ is faster than the modular exponentiation of 1024-bit RSA and the scalar multiplication of 160-bit ECC. However our implementation of η_T pairing does not sufficiently support the processing by exceptions, comparing with the release version of Bouncy Castle provider. Therefore the timing of η_T pairing with $\mathbb{F}_{3^{97}}$ is relatively fast, but η_T pairing is still calculated enough efficient. For the comparison we also show the timings of executing the same programs on a Pentium M 1.73GHz with 1GB RAM using J2SE.

5 Conclusion

In this paper, we presented the first implementation of Tate pairing over a mobile phone using Java. The η_T pairing over finite fields with characteristic three $\mathbb{F}_{3^m}$, which is the fastest version of Duursma-Lee algorithm, was implemented. There is no mathematical library from the Java cryptographic extension (JCE) for computing the arithmetic of finite fields $\mathbb{F}_{3^m}$, so that we implemented it from scratch in JAVA. The optimized implementation of the η_T pairing with $m = 97$ achieves about 0.5 seconds on a mobile phone FOMA SH901iS, NTT DoCoMo. This mobile phone is not the currently newest one, and the processing speed of the next models should become faster. Therefore the paring-based cryptosystems can be efficiently implemented on mobile phones using Java.

Acknowledgements

This research was supported by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

References

1. P. Barreto, S. Galbraith and C. O'hEigeartaigh, "Efficient pairing computation on supersingular abelian varieties", To appear in Designs, Codes, and Cryptography.
2. P. Barreto, H. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems", CRYPTO 2002, LNCS 2442, pp.354-368, 2002.
3. P. Barreto, B. Lynn, and M. Scott, "A note on efficient computation of cube roots in characteristic 3", IACR ePrint Archive, Report 2004/305, 2004.
4. G. Bertoni, J. Guajardo, S. Kumar, G. Orland, C. Paar, and T. Wollinger, "Efficient GF(p^m) arithmetic architectures for cryptographic application", CT-RSA 2003, LNCS 2612, pp.158-175, 2003.
5. D. Boneh, and M. Franklin , "Identity based encryption from the Weil pairing", SIAM J. Comput., vol.32, no.3, pp.586-615, 2001.
6. 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.
7. D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing", ASIACRYPTO 2001, LNCS 2248, pp.514-532, 2001.
8. I. Duursma and H. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$", ASIACRYPTO 2003, LNCS 2894, pp.111-123, 2003.
9. FOMA SH901iS, NTT DoCoMo. http://www.nttdocomo.co.jp/english/product/foma/
10. D. Hankerson, A. Menezes and S. Vanstone, *Guide to elliptic curve cryptography*, Springer-Verlag, 2004.
11. K. Harrison, D. Page, and N. Smart, "Software implementation of finite fields of characteristic three, for use in pairing-based cryptosystems", LMS J. Comput. Math., vol.5, pp.181-193, 2002.
12. Java 2 Platform, Micro Edition (J2ME). http://java.sun.com/javame/
13. Java Cryptography Extension (JCE). http://java.sun.com/products/jce/
14. T. Kerins, W. Marnane, E. Popovici, P. Barreto, "Efficient hardware for the Tate pairing calculation in characteristic three", CHES 2005, LNCS 3659, pp.412-426, 2002.
15. S. Kwon, "Efficient Tate pairing computation for supersingular elliptic curves over binary fields", IACR ePrint Archive, Report 2004/303.
16. M. Scott, N. Costigan, W. Abdulwahab, "Implementing Cryptographic Pairings on Smartcards", IACR ePrint Archive, Report 2006/144, 2006.
17. J. Silverman, *The arithmetic of elliptic curves*, Springer-Verlag, 1986.
18. T. Takagi, D. Reis, Jr., S.-M. Yen, B.-C. Wu, "Radix-r non-adjacent form and its application to pairing-based cryptosystem", IEICE Transactions, Vol.E89-A, No.1, pp.115-123, 2006.
19. S. Tillich and J. Großschadl, "A survey of public-key cryptography on J2ME-enabled mobile devices", ISCIS 2004, LNCS 3280, pp.935-944, 2004.
20. Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters", Version 1.0, 2000.
21. The Legion of the Bouncy Castle, Bouncy Castle Crypto APIs. http://www.bouncycastle.org/
22. Institute for Applied Information Processing and Communication, Sifting Secure Information and Communication Technologies. http://www.iaik.tugraz.at/

A Extended Euclidean Algorithm by Ternary Polynomial

The extended Euclidean algorithm for the polynomials over $\mathbb{F}_2[x]$ is shown [10]. We develop a ternary version of the extended Euclidean algorithm. Let $A(x)$

be the element in $\mathbb{F}_{3^m}$, and $deg()$ be the function computed degree. Inversion $(A(x)^{-1})$ mod $f(x)$ is computed as follows:

Algorithm 4 Inversion in $\mathbb{F}_{3^m}$

```
Input: $A(x) \in \mathbb{F}_{3^m} = \mathbb{F}_3[x]/(f(x))$
Output: $(A(x)^{-1}) \bmod f(x)$
 1: $u \leftarrow A(x),\ v \leftarrow f(x)$
 2: $g_1 \leftarrow 1,\ g_2 \leftarrow 0$
 3: while $deg(u) \neq 0$ do
 4:     $j \leftarrow deg(u) - deg(v)$
 5:     if $j < 0$ then
 6:         $u \leftrightarrow v,\ g_1 \leftrightarrow g_2,\ j \leftarrow -j$
 7:     end if
 8:     if $u_i + v_i \neq 0$ then
 9:         $v \leftarrow -v,\ g_2 \leftarrow -g_2$
10:     end if
11:     $u \leftarrow u + v \cdot x^j$
12:     $g_1 \leftarrow g_1 + g_2 \cdot x^j$
13: end while
14: if $u_i = 2$ then
15:     $g_1 \leftarrow -g_1$
16: end if
17: return $g_1$
```

B Arithmetic in $\mathbb{F}_{3^{3m}}$

Extension field $\mathbb{F}_{3^{3m}}$ can be represented by $\mathbb{F}_{3^m}[\rho]/g(\rho)$, where $g(\rho)$ is an irreducible polynomial $g(\rho) = \rho^3 - \rho - 1$ over $\mathbb{F}_{3^m}[X]$. We denote by $A(\rho)$ the element in $\mathbb{F}_{3^{3m}}$, where $A(\rho) = a_2\rho^2 + a_1\rho + a_0$ for $a_0, a_1, a_2 \in \mathbb{F}_{3^m}$. Arithmetic in field $\mathbb{F}_{3^{3m}}$ is performed as follows:

- **Addition :** $A(\rho) + B(\rho) = (a_2 + b_2)\rho^2 + (a_1 + b_1)\rho + (a_0 + b_0)$.
- **Multiplication :** Let $t_{00} = a_0b_0,\ t_{11} = a_1b_1,\ t_{22} = a_2b_2,\ t_{01} = (a_0 + a_1)(b_0 + b_1),\ t_{12} = (a_1 + a_2)(b_1 + b_2),\ t_{02} = (a_0 + a_2)(b_0 + b_2)$. Then $A(\rho)B(\rho) = (t_{02}+t_{11}-t_{00})+(t_{12}+t_{01}+t_{00}-t_{11})\rho+(t_{12}+t_{00}-t_{11}-t_{22})\rho^2$. Multiplication in $\mathbb{F}_{3^{3m}}$ is computed by 6 multiplications and 14 additions in $\mathbb{F}_{3^m}$.
- **Cube :** $A(\rho)^3 = a_2^3\rho^2 + (a_1^3 - a_2^3)\rho + (a_0^3 + a_1^3 + a_2^3)$. Cube in $\mathbb{F}_{3^{3m}}$ uses 3 cubes and 3 additions in $\mathbb{F}_{3^m}$.
- **Inversion :** We implemented using the algorithm shown by Kerins et. al. [14]. This algorithm can be obtained with 1 inversion, 9 multiplications and 11 additions in $\mathbb{F}_{3^m}$.