This paper provides an efficient algorithm for computing the $\eta_\ell$ pairing on supersingular elliptic curves over fields of characteristic two. In the proposed algorithm, we deploy a modified multiplication in $F_{2^n}$ using the Vandermonde matrix. For $F, G \in F_{2^n}$, the proposed multiplication method computes $\beta \cdot F \cdot G$ instead of $F \cdot G$ with some $\beta \in F_{2^n}^*$ because $\beta$ is eliminated by the final exponentiation of the $\eta_\ell$ pairing computation. The proposed multiplication method asymptotically requires only 7 multiplications in $F_{2^n}$ as $n \to \infty$, while the cost of the previously fastest Karatsuba method is 9 multiplications in $F_{2^n}$. Consequently, the cost of the $\eta_\ell$ pairing computation is reduced by 14.3%.

Keywords: Cryptography, $\eta_\ell$ pairing, finite field, loop unrolling, Vandermonde matrix.
that with the conventional Karatsuba method.

This paper proposes a faster algorithm for computing the $\eta_T$ pairing over $F_{2^n}$ using loop unrolling and the improved extension field multiplication in $F_{2^{kn}}$ with the Vandermonde matrix. We may compute $\beta \cdot F \cdot G$ instead of $F \cdot G$ for any $F, G \in F_{2^{kn}}$, and $\beta \in F_{2^n}$, in an algorithm for computing the $\eta_T$ pairing because $\beta$ powers to 1 in the final exponentiation, and has no effect on the pairing value. Therefore, the improved multiplication method computes the multiplication $\beta \cdot F \cdot G$ . The cost of the improved multiplication method is asymptotically $7$ multiplications in $F_{2^n}$ as $n \to \infty$. Note that a multiplication $F \cdot G$ requires $9$ multiplications in $F_{2^n}$ with the Karatsuba method. Combining the loop unrolling technique and the proposed multiplication method, we can achieve an efficient algorithm for computing the $\eta_T$ pairing.

The remainder of this paper is organized as follows. In section II, we provide a loop unrolling algorithm for computing the $\eta_T$ pairing over $F_{2^n}$. In section III, we describe polynomial multiplication. In section IV, we propose a multiplication method in $F_{2^{kn}}$ for the $\eta_T$ pairing and estimate the cost. In section V, we apply the proposed method to an algorithm for computing the $\eta_T$ pairing. Last, in section VI, we conclude this paper.

II. $\eta_T$ Pairing in Characteristic Two

A pairing $e$ used in cryptography such as the $\eta_T$ pairing, the Tate pairing, and the Ate pairing [5] are forms of bilinear mapping,

$$e : G_1 \times G_2 \rightarrow F_{p^k}^*,$$

where $G_1$ is a group based on $E(F_{p^k})$, $G_2$ is a group based on $E(F_{p^k})$ or $E(F_{p^k})$, $E$ is an elliptic curve defined over $F_{p^k}$, and $k$ denotes the embedding degree. Table 1 shows the relationships among pairings, finite fields, and embedding degrees.

In this paper, we deal with the $\eta_T$ pairing defined on supersingular elliptic curve

$$E_p : Y^2 = X^3 + X + b, \ b \in \{0,1\},$$

over $F_{2^n}$, where $n$ is a large enough prime for security of the $\eta_T$ pairing. Then, $n$ is an odd integer. The $\eta_T$ pairing is a bilinear mapping

$$\eta_T : E^p(F_{2^n}) \times E^p(F_{2^n}) \rightarrow F_{2^n}^*,$$

and is defined as $\eta_T(P,Q) = f_{2^n, p}(\psi(Q))^y$ for $P,Q \in E^p(F_{2^n})$, where $f_{2^n, p}$ is a function such that its divisor satisfies

$$\text{div}(f_{2^n, p}) = 2^n(p) - ((2^n - 1)(2^n - 2^{n+1} - 1)).$$

Here, $\psi$ is the distortion map defined as $\psi : (x,y) \rightarrow (x + s^2, y + sx + t)$, where $s^2s^2 + 1$ and $t = t + s$, and $W$ is an integer defined as $W = (2^n - 1)(2^n - 2^{n+1} - 1)$, where $e$ is a function defined as

$$e = \begin{cases} 1, & \text{if } n = 1,7 \pmod{8} \text{ and } b = 1 \\ 0, & \text{or } n = 3,5 \pmod{8} \text{ and } b = 0, \\ 1, & \text{otherwise.} \end{cases}$$

Algorithm 1. Original algorithm for $\eta_T$ pairing [6].

| Input: $P = (\alpha, \beta), Q = (\gamma, \beta) \in E^p(F_{2^n})$ |
| Output: $\eta_T(P,Q) \in F_{2^n}^*$ |
| 1: $w \leftarrow \alpha + \frac{n-1}{2}$ |
| 2: $F \leftarrow w \cdot (\gamma + \alpha + 1) + \beta + \frac{n+1}{2} + (w + \gamma)s + t$ |
| (cost: 6M) |
| 3: for $i = 0$ to $(n-1)/2$ do |
| 4: $w \leftarrow \alpha + \frac{n+1}{2}$ |
| 5: $F \leftarrow w \cdot \left( \gamma + \frac{n+1}{2} \right) + \beta + (w + \gamma)s + t$ |
| (G is ss-sparse, cost: M) |
| 6: $G \leftarrow F \cdot \eta_T$ (cost: 6M) |
| 7: if $i = (n-1)/2$ then |
| 8: $\alpha \leftarrow \sqrt{\beta}, \beta \leftarrow \sqrt{\gamma}, \gamma \leftarrow \gamma^2, \delta \leftarrow \delta^2$ (cost: 2R + 2S) |
| 9: end if |
| 10: end for |
| 11: return $F^W$ (cost: $F_6$) |

Algorithm 1 is efficient for computing the $\eta_T$ pairing [6]. In this algorithm, elements in $F_{2^{kn}}$ are represented with a basis $\{1,s,t,st\}$. This basis is called the $st$-basis, and we denote by $(a_0, a_1, a_2, a_3)$ the element $a_0 + a_1s + a_2t + a_3st$ in $F_{2^{kn}}$. For $a_0, a_1, a_2, a_3 \in F_{2^n}$, we also use another basis of $F_{2^{kn}}$, namely, the $z$-basis $\{1, z, z^2\}$ defined by the irreducible polynomial $h(z) = z^3 + z^2 + 1$ over $F_{2^n}$. We denote by $(b_0, b_1, b_2, b_3)$ the element $b_0 + b_1z + b_2z^2 + b_3 \in F_{2^{kn}}$. We can transform between

<table>
<thead>
<tr>
<th>Pairing</th>
<th>$\eta_T$ or Tate</th>
<th>Tate or Ate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>$F_{2^n}$</td>
<td>$F_{2^n}$</td>
</tr>
<tr>
<td>$k$</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 1. Embedding degree $k$ for pairings.
the st-basis and the z-basis virtually for free. Indeed, we have the following conversion algorithm for \((a_0, a_1, a_2, a_3) = (b_0, b_1, b_2, b_3)\):

\[
\begin{bmatrix}
 b_0 \\
 b_1 \\
 b_2 \\
 b_3 
\end{bmatrix} = \begin{bmatrix}
 1 & 0 & 0 & 0 \\
 0 & 1 & 1 & 0 \\
 0 & 0 & 1 & 1 \\
 0 & 0 & 0 & 1 
\end{bmatrix} \begin{bmatrix}
 a_0 \\
 a_1 \\
 a_2 \\
 a_3 
\end{bmatrix}
\]  

(1)

The notations used in algorithm 1 are defined as follows:

\[
\varepsilon_m = \begin{cases} 
0, & \text{if } m = 0, 1 \pmod 4 \\
1, & \text{otherwise,} 
\end{cases}
\]

\[ [m] = m \mod 2, \]

where \(n\) is the extension degree of \(F_{2^n}\) over \(F_2\), and \(m\) is an integer. Powering by \(W\) at step 11 in algorithm 1 is called the final exponentiation, which can be efficiently computed \([7]\).

We define symbols \(M, S, R, F\) as follows:

\[ M = \text{the cost of a multiplication in } F_{2^n}, \]

\[ S = \text{the cost of a squaring in } F_{2^n}, \]

\[ R = \text{the cost of a square root in } F_{2^n}, \]

\[ F = \text{the cost of the final exponentiation.} \]

Note that \(G\) at step 5 is the form of \((g_0, g_1, 1, 0)_0\) for \(g_0, g_1 \in F_{2^n}\), where \(g_0 = w \cdot (\gamma + [(n+1)/2]) + \delta + (\beta + b + \varepsilon_{n-1}/2)\) and \(g_1 = w + \gamma\). Such an element is called \(st\)-sparse in this paper. Note that the \(st\)-sparse element \((g_0, g_1, 1, 0)_0\) is converted to \((g_0, g_1+1, g_1, 0)_1\). We then call this element \(z\)-sparse. Note that there is no difference between the cost of multiplications using the \(st\)-basis and the cost of multiplications using the \(z\)-basis due to \((1)\). Now we distinguish among three multiplications in \(F_{2^n}\):

(a) random element by random element: \(\times\)

(b) random element by \((st, z)\)-sparse element: \(\times\)

(c) \((st, z)\)-sparse element by \((st, z)\)-sparse element: \(\times\)

where the random element is the form of \((g_0, g_1, g_2, g_3)_a\) or \((g_0, g_1, g_2, g_3)_b\) for \(g_0, g_1, g_2, g_3 \in F_{2^n}\). We use "\(\times\)" for multiplication of elements in \(F_{2^n}\) and polynomials with coefficients in \(F_{2^n}\).

The cost of each multiplication is shown in Table 2. Multiplication (a) is computed by the Karatsuba method, and the computation of (b) is proposed in \([1]\). We show that (c) can be computed with the cost of 3\(M\) in the appendix.

Now, we estimate the cost (the number of multiplications) of algorithm 1. The cost of each step is written at the end of each step. For example, step 2 takes \(M\) of "\((\gamma + a + 1)\)". Steps 1 and 2 are the initialization, with a cost of \(M\). Steps 3 to 8 are the main loop. Note that the number of iterations of the main loop, except from steps 3 to 6, is \((n+1)/2\), and the number of iterations of steps 3 to 6 is \((n-1)/2\), where \(n\) is an odd integer. Then, the cost of the main loop is \((n+1)/2 \cdot 7M + (n-1)/2 \cdot (2R + 2S)\). Step 11 is the final exponentiation, with a cost of \(F\). Therefore, the total cost of algorithm 1 is \((3.5n+4.5)M + (n-1)S + (n-1)R + F\).

1. Loop Unrolling Technique

We might be able to enhance the speed of the \(\eta_t\) pairing by unrolling the loop. Indeed, the unrolling technique is effective in the case of characteristic three \([2],[8]\).

Algorithm 2 is the loop unrolling algorithm for computing the \(\eta_t\) pairing over \(F_{2^n}\), based on algorithm 1. Algorithm 2 merges with algorithm 1 such that \(F \mapsto F \times G\) at step 6 in the \(2i\)-th and \((2i+1)\)-th iterations of algorithm 1 corresponds to \(G \mapsto G_0 \times G_t\) at step 9 and \(F \mapsto F \times G\) at step 10 in the \(i\)-th iteration of algorithm 2.

Note that "\(F \mapsto F \times G\)" of algorithm 1 takes 6\(M\) because \(F\) is random and \(G\) is sparse. On the other hand, "\(G \mapsto G_0 \times G_t\)" and "\(F \mapsto F \times G\)" of algorithm 2 take \(3M\) and \(9M\), respectively, because \(G_0\) and \(G_t\) are sparse and \(F\) is random. The number of iterations of algorithm 1 is twice that of algorithm 2. The cost of algorithm 2 is the same as that of algorithm 1, that is, \((3.5n+4.5)M + (n-1)S + F\), because \(6M+6M=3M+9M\).

Also note that if there was a multiplication method for \(F \times G\), namely, random element by random element, with a cost less than \(9M\), algorithm 2 would be more efficient than algorithm 1. Our goal is to find such a method.

III. Polynomial Multiplication

Multiplication in a finite field consists of polynomial multiplication and reduction modulo an irreducible polynomial.

| Multiplication symbols in \(F_{2^n}\) and cost. |
|---|---|---|
| \(a \times b\) | Random | Random | \(9M\) |
| \(a \times b\) | Random | \((st, z)\)-sparse | \(6M\) \([1]\) |
| \((st, z)\)-sparse \(b\) | \((st, z)\)-sparse | \((st, z)\)-sparse | \(3M\) |
Let \( a(z), b(z) \) be polynomials of degree \( k-1 \) with coefficients with \( F \). We consider polynomial multiplications of \( c(z)=a(z)\cdot b(z) \) and denote them by

\[
\begin{align*}
a(z) &= a_0 + a_1 z + a_2 z^2 + \cdots + a_{k-1} z^{k-1}, \\
b(z) &= b_0 + b_1 z + b_2 z^2 + \cdots + b_{k-1} z^{k-1}, \\
c(z) &= c_0 + c_1 z + c_2 z^2 + \cdots + c_{2k-2} z^{2k-2}.
\end{align*}
\tag{2}
\]

Note that \( c_{0}\equiv a_0 b_0 \) and \( c_{2k-2}\equiv a_{k-1} b_{k-1} \).

1. VM Method

Substituting \( z=\alpha \) in (2), we obtain

\[
\begin{align*}
a(\alpha) \cdot b(\alpha) &= c(\alpha) = c_0 + c_1 \alpha + c_2 \alpha^2 + \cdots + c_{2k-2} \alpha^{2k-2}
\end{align*}
\]

for any \( \alpha \in F \); thus,

\[
\begin{align*}
\begin{pmatrix}
\alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{2k-2}
\end{pmatrix}
\begin{pmatrix}
\alpha_0 & \alpha_1 & \ldots & \alpha_{2k-2} \\
1 & \alpha_1 & \ldots & \alpha_{2k-2} \\
\vdots & \vdots & \ddots & \vdots \\
1 & \alpha_{2k-2} & \ldots & \alpha_{2k-2}
\end{pmatrix}
&= 
\begin{pmatrix}
c_0 \\ c_1 \\ \vdots \\ c_{2k-2}
\end{pmatrix}
\begin{pmatrix}
a(\alpha_0) \cdot b(\alpha_0) \\ a(\alpha_1) \cdot b(\alpha_1) \\ \vdots \\ a(\alpha_{2k-2}) \cdot b(\alpha_{2k-2})
\end{pmatrix}
\end{align*}
\tag{3}
\]

holds for \( 2k-1 \) elements \( \alpha_0, \alpha_1, \ldots, \alpha_{2k-2} \in F \). The matrix \( A_V \) in (3) is called the VM. If this matrix is regular, then we can compute the vector \( C_V \), namely, \( c(z) \), by

\[
B_V = A_V^{-1} \cdot C_V.
\tag{4}
\]

2. DFT Method

We assume that \( F \) contains the primitive \((2k-1)\)th root of unity. Let \( \xi \) be this root of unity in \( F \). By setting \( \alpha_i = \xi^i \) in (4), we obtain

\[
\begin{vmatrix}
c_0 \\ c_1 \\ \vdots \\ c_{2k-2}
\end{vmatrix}
\begin{vmatrix}
1 & 1 & \cdots & 1 \\
1 & \xi & \cdots & \xi \\
\vdots & \vdots & \ddots & \vdots \\
1 & \xi^{2k-2} & \cdots & \xi^{2k-2}
\end{vmatrix}
\begin{pmatrix}
a(1) \cdot b(1) \\ a(\xi) \cdot b(\xi) \\ \vdots \\ a(\xi^{2k-2}) \cdot b(\xi^{2k-2})
\end{pmatrix}.
\]

The computation of \( c(z)=a(z)\cdot b(z) \) by the above matrix is called the DFT method.

Here, \( c(z) \) is computed with \((2k-1)\) multiplications \( a(1) \cdot b(1), a(\xi) \cdot b(\xi), \ldots, a(\xi^{2k-2}) \cdot b(\xi^{2k-2}) \) and one division by \( 2k-1 \) in \( F \) under the following assumptions:

(i) The cost of multiplication of \( \xi^i \) by each element in \( F \) is virtually zero,

(ii) The cost of computations of \( a(\xi^i) \) and \( b(\xi^i) \) with \( 0 \leq i, j \leq 2k-2 \) is virtually zero, and

(iii) \((2k-1)\) is not equal to the characteristic of \( F \).
However, it seems hard to satisfy all three conditions for many finite fields, especially the finite fields for the \( \eta_t \) pairing (see the next sections).

3. GPS Multiplication for \( F_{y^t} \) [3]

In the case of characteristic three, the algorithm for computing the \( \eta_t \) pairing includes computations of multiplications in \( F_{y^t} \), where \( n \) is a prime. Unfortunately, the DFT method cannot be used for multiplications in field \( F_{y^t} \) because the 1st (11=2×6-1) primitive root of unity is not contained in \( F_{y^t} \).

The GPS multiplication tries to efficiently compute a multiplication in \( F_{y^t} \) by modifying the DFT method in the third extension field \( F_{y^t} \) over \( F_{y^t} \). Now we consider a polynomial multiplication with coefficients in \( F_{y^t} \) of degree 2, that is,

\[
(a_0 + a_1 z + a_2 z^2) \cdot (b_0 + b_1 z + b_2 z^2) = c_0 + c_1 z + c_2 z^2 + c_3 z^3 + c_4 z^4.
\]

(5)

Elements in \( F_{y^t} \) are represented as \( \{u_0 + u_1 \sigma, u_0, u_1 \in F_{y^t}\} \), where \( \sigma^2 = -1 \) and \( \sigma \) is the fourth primitive root of unity in \( F_{y^t} \). Then we obtain

\[
(a_0 + a_1 \sigma^i + a_2 (\sigma^i)^2) \cdot (b_0 + b_1 \sigma + b_2 (\sigma^i)^2) - c_4 (\sigma^i)^4
dot (\sigma^i)^4
\]

by setting \( z = \sigma^i \) in (5) for \( i = 0, 1, 2, 3 \). They yield the following matrix equation:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & \sigma & \sigma^2 & \sigma^3 \\
1 & \sigma^2 & \sigma^4 & \sigma^6 \\
1 & \sigma^3 & \sigma^6 & \sigma^9
\end{bmatrix}
\begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
c_3
\end{bmatrix}
= \begin{bmatrix}
(\alpha(1) \cdot b(1) - c_4) \\
\alpha(\sigma) \cdot b(\sigma) - c_4 \\
\alpha(\sigma^2) \cdot b(\sigma^2) - c_4 \\
\alpha(\sigma^3) \cdot b(\sigma^3) - c_4
\end{bmatrix}.
\]

(6)

Because \( \sigma^2 = -1 \) and \( c_4 = a_2, b_2 \), (6) can be eventually converted as follows:

\[
\begin{bmatrix}
c_0 \\
c_1 \\
c_2 \\
c_3
\end{bmatrix}
= \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & -\sigma & -1 & \sigma \\
1 & -1 & 1 & -1 \\
1 & \sigma & -1 & -\sigma
\end{bmatrix}
\begin{bmatrix}
(\alpha(1) \cdot b(1) - a_2) \\
(\alpha(\sigma) \cdot b(\sigma) - a_2) \\
(\alpha(\sigma^2) \cdot b(\sigma^2) - a_2) \\
(\alpha(\sigma^3) \cdot b(\sigma^3) - a_2)
\end{bmatrix}.
\]

(7)

Note that the cost of multiplication of \( \sigma \) by any element in \( F_{y^t} \) is virtually zero because \( (u_0 + u_1 \sigma) \cdot \sigma = -u_0 + u_1 \sigma \), where \( u_0 + u_1 \sigma \in F_{y^t}, (u_0, u_1 \in F_{y^t}) \). Therefore, the cost of computing (7) is 5 multiplications in \( F_{y^t} \). A multiplication in \( F_{y^t} \) can be computed with 15 multiplications in \( F_{y^t} \) by using the Karatsuba method for the multiplication in \( F_{y^t} \).

4. Application of DFT Method and GPS Multiplication to \( F_{y^t} \)

In the algorithm for computing the \( \eta_t \) pairing over \( F_{y^t} \), we need multiplications in \( F_{y^t} \). Here, we consider whether we can apply the DFT method and the GPS multiplication to finite field \( F_{y^t} \).

There is no 7th \((7 = 2 \times 4 - 1)\) primitive root of unity in \( F_{y^t} \), so we cannot apply the DFT method to \( F_{y^t} \).

We can apply the GPS multiplication to \( F_{y^t} \) because there is a 2nd \((2 = 2 \times 2 - 2)\) primitive root of unity in \( F_{y^t} \). Then, the cost of a multiplication in \( F_{y^t} \) by the GPS multiplication is equal to 3 \((=2 \times 2 - 1)\) multiplications in \( F_{y^t} \), that is, \( 9M \), where \( M \) means the cost of a multiplication in \( F_{y^t} \) as defined in section II. However, the Karatsuba method also provides a multiplication method with a cost of \( 9M \). Therefore, the GPS multiplication cannot reduce the cost in the case of characteristic two as it can in the case of characteristic three.

IV. Proposed Multiplication for \( \eta_t \) Pairing

In this section, we present the proposed multiplication method, which is an improvement of the VM method over \( F_{y^t} \). The main idea is to deploy the computations of \( \beta \cdot F \times G (\beta \in F_{y^t}) \) instead of \( F \times G \) in algorithm 2, where \( F, G \in F_{y^t} \). Note that \( \beta^6 = 1 \) holds, where powering by \( W \) is the final exponentiation of the \( \eta_t \) pairing; thus, we can deal with the relaxed multiplication \( \beta \cdot F \times G \) in the following subsection.

1. Main Idea

Here we discuss how to modify the multiplication \( F \times G \) at step 10 in algorithm 2, where \( F, G \in F_{y^t} \). Let us use the z-basis of \( F_{y^t} \) over \( F_{y^t} \), which has a definition polynomial of \( h(z) = z^2 + z + 1 \). Let \( F(z) = f_0 + f_1 z + f_2 z^2 + f_3 z^3 \) and \( G(z) = g_0 + g_1 z + g_2 z^2 + g_3 z^3 \), for \( f_0, g_0, f_1, g_1, f_2, g_2, f_3, g_3 \in F_{y^t} \). To evaluate \( F \times G \), we try to compute polynomial

\[
e(z) = e_0 + e_1 z + e_2 z^2 + e_3 z^3 + e_4 z^4 + e_5 z^5 + e_6 z^6,
\]

(8)

where \( e(z) = (f(z), g(z)) \) and \( e(\eta_t) = h(z) \mod h(2) \).

First, we note that \( e_0 \) and \( e_1 \) can be easily computed, that is, \( e_0 = f_0 \cdot g_0 \) and \( e_1 = f_1 \cdot g_1 \). Next, we explain how to find the other coefficients \( e_2, e_3, e_4, e_5, e_6 \) using the VM method.

Let \( \{1, x, x^2\} \) be a polynomial basis of \( F_{y^t} \) over \( F_2 \). Substituting \( x = 1, x, x^2 \) for \( e(z) = (f(z), g(z)) \), we have the following relationship:
Then, the cost of computing \( A_{ij} \alpha = \beta \) as \( A_{ij} \) becomes \( A_{ij} \alpha \) as \( \beta \) becomes \( \alpha \).

Therefore, dividing by any element \( e \), we eventually obtain a new formula computing \( e(z) = \beta \cdot f(z) \cdot g(z) \) as follows:

\[
\begin{pmatrix}
\alpha_0 = x^0, \\
\alpha_1 = x^1, \\
\alpha_2 = x^2, \\
\alpha_3 = x^3, \\
\alpha_4 = x^4
\end{pmatrix} = \begin{pmatrix}
\beta_0 = x^0, \\
\beta_1 = x^1, \\
\beta_2 = x^2, \\
\beta_3 = x^3, \\
\beta_4 = x^4
\end{pmatrix}
\]

Hence, the cost of the reduction \( e(z) \mod f(z) \) is virtually zero. It is then enough to consider the cost of computing \( (9) \), that is, \( e(z) = \beta \cdot f(z) \cdot g(z) \).

Next, we discuss the cost of computing \( e(z) = \beta \cdot f(z) \cdot g(z) \). To estimate the cost of the proposed multiplication method, we have to consider the multiplication with polynomials of very small degrees. Define \( M \) as

\[
\frac{M}{n-1}
\]

where \( M \) is the cost of a multiplication in \( F_{2^n} \) as defined in section II. We assume the following cost for the multiplication with polynomials of very small degrees.

**Assumption 1.** Let \( a \) be a random element in \( F_{2^n} \), and let \( b \) be an element of degree \( d_b \) in \( F_{2^n} \). Then, the cost of a multiplication \( a \cdot b \) in \( F_{2^n} \) becomes \( d_b M \).

Although the cost of multiplication depends on the implementation of a multiplication, assumption 1 is correct when the multiplication is implemented by the shift-addition method \([11]\) or its variations using window methods \([4, 11, 12]\), which are basic multiplication methods. We can explain how assumption 1 holds.

First, we consider a multiplication of polynomials \( a(x) \cdot b(x) \) in \( F_{2^n}[x] \) by the shift-addition method or its variations using window methods. The basic step of their algorithms is to shift \( a(x) \) from right to left (or left to right) and perform addition \( a(x) + b(x) \) based on each coefficient of \( b(x) \). The shift-addition method requires \( d_x d_y \) shifts and \( 0.5d_x d_y \) additions in \( F_{2^n} \), where \( d_x \) and \( d_y \) are the degree of \( a(x) \), and the degree of \( b(x) \), respectively.

Note that after computation of \( (9) \), we need a computation \((e_0 + e_1 z + e_2 z^2 + e_3 z^3 + e_4 z^4) \mod h(z)\), where \( h(z) = z^2 + z + 1 \).

The proposed multiplication method consists of computing \( e(z) = \beta \cdot f(z) \cdot g(z) \) using \( (9) \) and the reduction \( e(z) \mod h(z) \).

First we deal with the cost of the reduction modulo \( h(z) = z^2 + z + 1 \). This modulo is the following linear transform:

\[
(e_0 + e_1 z + e_2 z^2 + e_3 z^3 + e_4 z^4) \mod h(z)
\]

\[
= (e_0 + e_1 z + e_2 z^2 + e_3 z^3 + e_4 z^4) \mod h(z)
\]

where \( e_0, e_1, e_2, e_3, e_4 \) are the coefficients of \( e(z) \cdot f(z) \cdot g(z) \) and \( h(z) = z^2 + z + 1 \).
respectively. The number of shifts and additions in the variations of the shift-addition method using the window methods of width \( w \) requires \( O(d_s2^w) \) in a precomputation stage and \( O(d_s/d_\nu) \) in the main loop. The constant term of \( O \) notation depends on the using algorithm. Therefore, the ratio of \( a(x) \cdot h(x) \) in \( F_2[x] \) of \( d_s=d_\nu \) over \( d_s=d_\nu \) is equal to \( \frac{d_\nu}{d_s} \). Then, we have

\[
\text{the cost of } \frac{a(x) \cdot h(x)}{d_s} = \frac{d_\nu}{d_s} \mu, \quad \text{where } \mu \text{ is the cost of a multiplication } a(x) \cdot h(x) \text{ in } F_2[x] \text{ of } d_s=d_\nu.
\]

Next, using (10), we show assumption 1 is true. In general, a random element has degree \((n-1)\) and \( d_s \leq n-1 \). Note that \( M \) is the same as \( \mu \) in (10) because the cost of reduction modulo, \( h(x) \), is virtually zero. Setting \( d_\nu=n-1 \), we have the cost of \( a(x) \cdot h(x) \) becomes \( d_s/(n-1)=\mu=M \). This completes the explanation of assumption 1.

Now, we consider the cost of (9). First, we compute the components of the vector \( C_p \). Then, we compute a matrix multiplication \((a_0) \cdot C_p \). Finally, we compute \( e_0 \) and \( e_6 \).

A. Computation of Components of \( C_p \)

To compute the components of the vector \( C_p \), we may compute the following sets:

\[
\begin{align*}
\{ f(1), f(x), f(x+1), f(x^2), f((x+1)^2) \}, \\
\{ g(1), g(x), g(x+1), g(x^2), g((x+1)^2) \}, \\
\{ f(1) \cdot g(1), f(x) \cdot g(x), f(x+1) \cdot g(x+1), f(x^2) \cdot g(x^2), f((x+1)^2) \cdot g((x+1)^2) \}, \\
\{ f_1 \cdot g_0, f_2 \cdot g_0, x^2 \cdot g_0, (x+1)^2 \cdot g_0, f_1 \cdot g_1 \cdot x^2, f_2 \cdot g_1 \cdot x^2, f_1 \cdot g_2 \cdot (x+1)^2, f_2 \cdot g_2 \cdot (x+1)^2 \}, \\
\{ f_0, g_0 \}.
\end{align*}
\]

We can compute set (i) using algorithm 3, the cost of which is \( 16M \) due to assumption 1. We can also compute set (ii) using algorithm 3. Set (iii) needs \( 5M \). We can compute set (iv) using algorithm 4, which costs \( 12M+5M \). Finally, we compute \( f_0 \cdot g_0 \) which costs \( M \). Therefore, we take \( 7M+44M \) to compute the components of \( C_p \).

B. Computation of Matrix Multiplication \((a_0) \cdot C_p \)

Suppose \( C_p \) is represented as \((C_0, C_1, C_2, C_3, C_4)^T \) with \( C_j \in F_2 \) for \( 0 \leq j \leq 4 \). Then, to compute \((a_0) \cdot C_p \), we may compute \( a_j \cdot C_i \) for \( 0 \leq i, j \leq 4 \). Here, we consider the computation of set \( \{(a_0) \cdot C_i \} \) \( 0 \leq i \leq 4 \). Note that the set is obtained from \( x^i C_0 \) with \( 0 \leq j \leq 8 \), and this is the maximum degree of \( a_0 \). Therefore, the cost of \((a_0) \cdot C_0 \) is \( 8M \). Similarly, the costs of computing sets \( \{(a_0) \cdot C_1 \}, \{(a_0) \cdot C_2 \}, \{(a_0) \cdot C_3 \}, \) and \( \{(a_0) \cdot C_4 \}, \) are \( 7M, 7M, 4M, \) and \( 4M \), respectively. Therefore, we take \( 8M+7M+7M+4M+4M = 30M \) to compute \((a_0) \cdot C_p \).

Algorithm 3. Computation of \((f_{(1)}, f_{(x)}, f_{(x+1)}, f_{(x^2)}, f_{((x+1)^2)})\)

Input: \( f(x) = f_0 + f_1 x + f_2 x^2 + f_3 x^3 + f_4 x^4 \) \( \in F_2[x], \) \( x \in F_2^* \) such that \( x \) forms the basis of \( F_2^* \) over \( F_2 \).

Output: \( \{(f_{(1)}), f_{(x)}, f_{(x+1)}, f_{(x^2)}, f_{((x+1)^2)}\} \)

1: \( T_0 \leftarrow f_0, T_1 \leftarrow f_1, T_2 \leftarrow f_2, T_3 \leftarrow f_3 \)
2: \( R_0 \leftarrow T_0 + T_1 + T_2 + T_3 \)
3: \( T_1 \leftarrow T_0 \cdot x \) (cost: \( \frac{9M}{2} \))
4: \( T_2 \leftarrow T_1 \cdot x^2 \) (cost: \( \frac{2M}{2} \))
5: \( T_3 \leftarrow T_1 \cdot x^3 \) (cost: \( \frac{3M}{2} \))
6: \( R_1 \leftarrow T_0 + T_1 + T_2 + T_3 \)
7: \( T_2 \leftarrow T_1 \cdot x \) (cost: \( \frac{9M}{2} \))
8: \( T_3 \leftarrow T_1 \cdot x^2 \) (cost: \( \frac{2M}{2} \))
9: \( T_4 \leftarrow T_1 \cdot x^3 \) (cost: \( \frac{3M}{2} \))
10: \( R_2 \leftarrow R_1 + f_1 + f_2 + T_0 \)
11: \( T_0 \leftarrow f_0 \cdot (x^2 + x + 1) \) (cost: \( \frac{2M}{2} \))
12: \( R_3 \leftarrow R_2 + f_1 + f_2 + f_3 + T_0 \)
13: \( T_0 \leftarrow T_0 \cdot (x^2 + x + 1) \) (cost: \( \frac{2M}{2} \))
14: \( R_4 \leftarrow R_3 + f_1 + f_2 + f_3 + T_0 \)
15: return \{\( R_0, R_1, R_2, R_3, R_4 \)\} (total cost: \( 16M \))

C. Computation of \( e_0 \) and \( e_6 \)

We compute \( e_0 = \beta \cdot (f_0 \cdot g_0) \) and \( e_6 = \beta \cdot f_6 \cdot g_6 \). Recall that \( f_0 \cdot g_0 \) and \( f_6 \cdot g_6 \) were already computed when we computed the components of \( C_p \). Thus, the computation of \( e_0 \) and \( e_6 \) takes \( 16M \) because the degree of \( \beta \) is 8.

D. Total Cost of Proposed Multiplication Method

From sections IV.2.A, IV.2.B, and IV.2.C, we can evaluate (9) with \( 7M+90M=\left(7+\frac{90}{(n-1)}\right)M \). The cost is asymptotically equal to \( 7M \) as \( n \rightarrow \infty \) and is \( 7.38M \) in the case of \( n=239 \). On the other hand, the previous multiplication using the Karatsuba method requires \( 9M \) for any degree \( n \). We summarize these estimations in Table 3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Any ( n )</th>
<th>( n = 239 )</th>
<th>( n \rightarrow \infty )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karatsuba method</td>
<td>9M</td>
<td>9M</td>
<td>9M</td>
</tr>
<tr>
<td>Proposed multiplication</td>
<td>( \left(7+\frac{90}{n-1}\right)M )</td>
<td>7.38M</td>
<td>7M</td>
</tr>
</tbody>
</table>

Table 3. Computation of cost of \( F \equiv F \times G \).
Algorithm 4. Computation of $f_i \cdot g_j \cdot \cdots \cdot f_i \cdot g_j \cdot (x+1)^{12}$

Input: $f_i, g_j, x \in F_{p^2}$ such that $x$ forms the basis of $F_{p^2}$ over $F_2$

Output: $\{f_i \cdot g_j \cdot f_i \cdot g_j \cdot (x+1)^4, f_i \cdot g_j \cdot f_i \cdot g_j \cdot (x+1)^4\}$

1. $T \leftarrow f_i \cdot g_j$ (cost: $M$)
2. $R_0 \leftarrow T, R_1 \leftarrow T, R_2 \leftarrow T$
3. $T \leftarrow T \cdot x^3$ (cost: $2M$)
4. $R_3 \leftarrow R_0 + T$
5. $T \leftarrow T \cdot x^2$ (cost: $2M$)
6. $R_3 \leftarrow R_2 + T$
7. $T \leftarrow T \cdot x^2$ (cost: $2M$)
8. $R_4 \leftarrow T, R_5 \leftarrow R_4 + T$
9. $T \leftarrow T \cdot x^4$ (cost: $4M$)
10. $R_6 \leftarrow T + R_4$
11. $T \leftarrow T \cdot x^4$ (cost: $4M$)
12. $R_7 \leftarrow T, R_8 \leftarrow R_7 + T$
13. return $\{R_0, R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8\}$ (total cost: $12M$)

3. Applying Proposed Multiplication to General Cases

In this subsection, we discuss applying the proposed multiplication to general cases. Recall that the proposed multiplication computes relaxed multiplication $\beta \cdot F \times G$ for $F, G \in F_{p^2}$, where $\beta \in F_{p^2}$ is an element. We easily make the general version of the proposed multiplication in any finite field $F_{p^2}$; thus, $C_{ij}$ in (9) becomes the $(2k-3)$th vector and $(\alpha_0)$ becomes a $(2k-3) \times (2k-3)$ matrix. Define $M_{p^2}$ as the cost of a multiplication in $F_{p^2}$.

Note that the proposed multiplication is effective in pairing computations because $\beta$ powers to 1 in the final exponentiation and has no effect on the pairing value. Then, we consider applying the proposed multiplication to other fields in Table 1.

A. Case of $(F_{p^2}, k=6)$

The $F_{p^2}$ version of the proposed multiplication takes $(11+\gamma/(n-1))M_{p^2}$, where $\gamma$ is a constant number depending on a vector $C_{ij}$ and a matrix $(\alpha_0)$ in (9). Therefore, the $F_{p^2}$ version of the proposed multiplication asymptotically takes $11M_{p^2}$ to compute a relaxed multiplication in $F_{p^2}$. However, if the Vandermonde matrix $A_{p^2}$ is not carefully constructed, then $\gamma$ seems to be very large, and the proposed multiplication incurs a greater cost than the GPS multiplication (see [3] or section III.3) which takes $15M_{p^2}$ in practical cases as $n$ is several hundred. Finding the vector $C_{ij}$ and the matrix $(\alpha_0)$ such that $A$ becomes smallest will be the subject of future work.

B. Case of $(F_{p^2}, k=any value)$

In this case components of $C_{ij}$ and $(\alpha_0)$ are elements in $F_{p^2}$ and computation of $(\alpha_0) \cdot C_{ij}$ in (9) in general takes $(2k-3)M_{p^2}$. Therefore, the $F_{p^2}$ version of the proposed multiplication takes more than $(2k-3)M_{p^2}$ which means the Karatsuba method is better than the $F_{p^2}$ version of the proposed method.

V. New Efficient Algorithm for $\eta_r$ Pairing

In this section, we present a new efficient algorithm for computing the $\eta_r$ pairing using the new multiplication method presented in section IV.

As we described in the last paragraph of section IV.2, if $F \times G$ (random×random) at step 10 in algorithm 2 is efficiently computed, we can obtain an algorithm for computing the $\eta_r$ pairing. Then, we propose an efficient multiplication method computing $\beta \cdot F \times G$ instead of $F \cdot G$ in section IV.4. Recall that $\beta$ powers to 1 in the final exponentiation, and therefore has no effect on the pairing value. Consequently, we can obtain a new algorithm for computing the $\eta_r$ pairing, namely, algorithm 5, applying the proposed multiplication method to algorithm 2.

Note that we need to use the $z$-basis, not the $s$-basis, to provide the new algorithm for the $\eta_r$ pairing. However, the conversion between the $s$-basis and the $z$-basis is virtually zero due to (1). In algorithm 5, steps 2 to 18 are executed using the $z$-basis, and the result of step 18 is converted to a value of the $s$-basis. The output of algorithm 5 is then the same as algorithm 2.

Next, we estimate algorithm 5 without the final exponentiation, which can be efficiently computed [7]. Note that the cost of each step of algorithm 5 except step 10 is the same as those of algorithm 2 due to (1). Let $m$ be the number of iterations of the main loop of algorithm 5. Then, the cost of algorithm 5 is calculated as:

\[(\text{the cost of algorithm 5}) = (\text{(the cost of algorithm 2)}) + m \cdot ((\text{the cost of step 9 in algorithm 5}) - (\text{the cost of step 9 in algorithm 2})) = (3.5n+4.5)M^{+}(n-1)S^{+}(n-1)R^{+} + m \cdot ((7+90(n-1))M^{+}9M)\]

because the cost of algorithm 2 is $(3.5n+4.5)M^{+}(n-1)S^{+}(n-1)R^{+}$ as described in section II, where $M, S, R$ are the cost of a multiplication, a squaring, and a square root in $F_{p^2}$, respectively. Note $m=(n-1)/4$ if $n=1 \pmod{4}$, and $m=(n+1)/4$ if $n=3 \pmod{4}$. Then, we have the cost of algorithm 5 as $(3.5n+24.5n-27.5)(n-1)M^{+}(n-1)R^{+}(n-1)S^{+}$ if $n=1 \pmod{4}$ and as $(3.5n+23.5n+18.5)(n-1)M^{+}(n-1)R^{+}(n-1)S^{+}$ if $n=3 \pmod{4}$. 

136 Masaaki Shirase et al.

ETRI Journal, Volume 31, Number 2, April 2009
Algorithm 5. Proposed algorithm for \( \eta_j \) pairing

Input: \( P = (\alpha, \beta), Q = (\gamma, \delta) \in \mathbb{F}^n(\mathbb{F}_2^n) \)

Output: \( \eta_j(P, Q) \in \mathbb{F}_2^n \)

1: \( w \leftarrow \alpha + \left[ \frac{n+1}{2} \right] \)
2: \( F \leftarrow w \cdot (\gamma + \alpha + 1) + \delta + (\beta + b + \epsilon_{i+1/2}) + (w + \gamma + 1)z + (w + \gamma)z^2 \) \( \text{(cost: } M) \)
3: for \( i = 0 \) to \( (n-3)/4 \) do
4: \( w \leftarrow \alpha + \left[ \frac{n+1}{2} \right] \)
5: \( G_w \leftarrow w \cdot \left( \gamma + \left[ \frac{n+1}{2} \right] \right) + \delta + (\beta + b + \epsilon_{i+1/2}) + (w + \gamma + 1)z + (w + \gamma)z^2 \) \( \text{(G_w is z-sparse, cost: } M) \)
6: \( \alpha \leftarrow \sqrt{\alpha}, \beta \leftarrow \sqrt{\beta}, \gamma \leftarrow \gamma^2, \delta \leftarrow \delta^2 \) \( \text{(cost: } 2R+2S) \)
7: \( w \leftarrow \alpha + \left[ \frac{n+1}{2} \right] \)
8: \( G_i \leftarrow w \cdot \left( \gamma + \left[ \frac{n+1}{2} \right] \right) + \delta + (\beta + b + \epsilon_{i+1/2}) + (w + \gamma + 1)z + (w + \gamma)z^2 \) \( \text{(G_i is z-sparse, cost: } M) \)
9: \( G \leftarrow G_0 \times G_i \) \( \text{(cost: } 3M) \)
10: \( F \leftarrow F \times G \), where \( \beta = x^1 + x^3 + x^4 + x^5 \) \( \text{(computed by (9), cost: by Table 3)} \)
11: \( \alpha \leftarrow \sqrt{\alpha}, \beta \leftarrow \sqrt{\beta}, \gamma \leftarrow \gamma^2, \delta \leftarrow \delta^2 \) \( \text{(cost: } 2R+2S) \)
12: end if
13: if \( i = 1 \) (mod 4) then
14: \( w \leftarrow \alpha + \left[ \frac{n+1}{2} \right] \)
15: \( G_w \leftarrow w \cdot \left( \gamma + \left[ \frac{n+1}{2} \right] \right) + \delta + (\beta + b + \epsilon_{i+1/2}) + (w + \gamma + 1)z + (w + \gamma)z^2 \) \( \text{(G_w is z-sparse, cost: } M) \)
16: \( F \leftarrow F \times G \) \( \text{(cost: } 6M) \)
17: end if
18: \( (f_0, f_1, f_2, f_3, f_4) \leftarrow F^w \) \( \text{(cost: } F_G) \)
19: return \( (f_0, f_2 + f_1, f_1 + f_2 + f_3, f_3) \)

We aim to estimate the costs of algorithms 1, 2, and 5 using only \( M \) to easily compare the proposed algorithm 5 with the conventional algorithms 1 and 2. We then need a relationship among \( M, S, \) and \( R \). Suppose that \( R = 0.5M \) as followed by Kwon [7]. Note that \( a^1/2 = a^{2^{-1}} \) for any \( a \in \mathbb{F}_2^n \) because \( a^{2^n} = a \). Therefore, \( (n + 1/2)S = R \), that is, \( S = 1/(2n - 4) \cdot M \). The comparison is shown in Table 4.

| Table 4. Cost comparison between algorithms 1 and 2 and algorithm 5. |
|---------------------------------|-----------------|----------------|
| \( n = 239 \)                   | \( n \rightarrow \infty \) |
| Algorithms 1 and 2             | 960.5\(M \)     | 4n\(M \)       |
| Proposed algorithm 5           | 863.2\(M \)     | 3.5n\(M \)       |

Therefore, the cost of computing the \( \eta_j \) pairing is reduced by 11.3% and 14.3% for \( n=239 \) and \( n \rightarrow \infty \), respectively.

VI. Conclusion

The loop unrolling technique and the GPS multiplication are effective in the \( \eta_j \) pairing on supersingular curve in characteristic three. However, they are ineffective in the \( \eta_j \) pairing in characteristic two. In the case of characteristic two, the loop unrolling technique changes \( (n+1)/2 \) multiplications of “sparse element by random element” in \( \mathbb{F}_2^n \), in the algorithm for computing the \( \eta_j \) pairing (algorithm 1) to \( (n+1)/4 \) “sparse by sparse” multiplications and \( (n+1)/4 \) “random by random” multiplications. Note that multiplications of a “sparse by random,” a “sparse by sparse,” and a “random by random” in \( \mathbb{F}_2^n \) take 6\(M \), 3\(M \), and 9\(M \), respectively, using the Karatsuba method, where \( M \) means the cost of a multiplication in \( \mathbb{F}_2^n \). The GPS multiplication is a method to compute a multiplication of “random by random,” and in the case of \( \mathbb{F}_2^n \), the GPS multiplication incurs the same cost as the Karatsuba method. Then, the loop unrolling technique and the GPS multiplication are ineffective in the \( \eta_j \) pairing in characteristic two.

We introduced a relaxed multiplication which computes \( \beta \cdot F \times G \) instead of \( F \times G \) for \( F, G \in \mathbb{F}_2^n \), where \( \beta \) is an element in \( \mathbb{F}_2^n \). The \( \beta \) powers to 1 in the final exponentiation
and therefore has no effect on the pairing value. The relaxed multiplication can be computed using the Vandermonde matrix, the inverse of which has coefficients of small degrees. The multiplication in $F_{2^n}$ can be computed by 7 multiplications in $F_2$ for $n \rightarrow \infty$, while the previously fastest Karatsuba method requires 9 multiplications. Consequently, the cost of the $\eta_T$ pairing computation is reduced by 14.3\% as $n \rightarrow \infty$.

Appendix. Sparse Multiplication

To describe the cost of (c) in Table 3, we provide the algorithms A1 and A2 for computing a multiplication of $(st, z)$-sparse elements in $F_{2^n}$.

**Algorithm A1.** Multiplication of $st$-sparse elements in $F_{2^n}$

Input: $st$-sparse $a=(a_0, a_1, 1, 0)_n$, $b=(b_0, b_1, 1, 0)_n \in F_{2^n}$

Output: $c=(c_0, c_1, c_2, c_3)_n \in F_{2^n}$

1: $T_0 \leftarrow a_0 \cdot b_0$ (cost: $M$)
2: $T_1 \leftarrow a_1 \cdot b_1$ (cost: $M$)
3: $T_2 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1)$ (cost: $M$)
4: $R_0 \leftarrow T_0 + T_1$
5: $R_1 \leftarrow T_0 + T_2 + 1$
6: $R_2 \leftarrow a_0 + b_1 + 1$
7: $R_3 \leftarrow a_1 + b_1$
8: return $(R_0, R_1, R_2, R_3)_n$

**Algorithm A2.** Multiplication of $z$-sparse elements in $F_{2^n}$

Input: $z$-sparse $a=(a_0, a_1+1, a_1, 0)_n$, $b=(b_0, b_1+1, b_1, 0)_n \in F_{2^n}$

Output: $c=(c_0, c_1, c_2, c_3)_n \in F_{2^n}$

1: $T_0 \leftarrow a_0 \cdot b_0$ (cost: $M$)
2: $T_1 \leftarrow a_1 \cdot b_1$ (cost: $M$)
3: $T_2 \leftarrow (a_0 + a_1) \cdot (b_0 + b_1)$ (cost: $M$)
4: $R_0 \leftarrow T_0 + T_1$
5: $R_1 \leftarrow T_0 + T_2 + a_0 + b_0$
6: $R_2 \leftarrow T_0 + T_2 + a_0 + b_1 + 1$
7: $R_3 \leftarrow a_1 + b_1$
8: return $(R_0, R_1, R_2, R_3)_n$

References


Masaaki Shirase received the BSc in mathematics from Ibaraki University in 1994, and MIS and DrIS degrees from Japan Advanced Institute of Science and Technology (JAIST) in 2003 and 2006, respectively. He is currently a research associate with the School of System Science Information at Future University-Hakodate. His research interests are algorithms and implementation of cryptography.

Tsuyoshi Takagi received the BSc and MSc degrees in mathematics from Nagoya University in 1993 and 1995, respectively. He engaged in research on network security at NTT Laboratories from 1995 to 2001. He received the Dr.rer.nat degree from Technische Universität Darmstadt in 2001. He was an assistant professor with the Department of Computer Science at Technische Universität Darmstadt until 2005. He is currently a professor with the School of Systems Information Science at Future University-Hakodate. His current research interests are information security and cryptography. Dr. Takagi is a member of International Association for Cryptologic Research (IACR).
Dooho Choi received his BS degree in mathematics from Sungkyunkwan University, Seoul, Korea in 1994, and the MS and PhD degrees in mathematics from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1996 and 2002, respectively. He has been a senior researcher with Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea since Jan. 2002. His research interests include security technologies of RFID and wireless sensor networks. He is an editor of the ITU-T X.1171 (X.nidsec-1).

DongGuk Han received his BS degree in mathematics from Korea University in 1999, and his MS degree in mathematics from Korea University in 2002. He received his PhD of engineering in Information Security from Korea University in 2005. He was a PostDoc with Future University Hakodate, Japan. After finishing the doctoral course, he was an exchange student with the Department of Computer Science and Communication Engineering, Kyushu University, Japan from April 2004 to March 2005. He was a senior researcher in Electronics and Telecommunications Research Institute (ETRI), Daejeon, Rep. of Korea. He is currently working as an assistant professor with the Department of Mathematics of Kookmin University, Seoul, Rep. of Korea. He is a member of KIISC, IEEK, and IACR.

Howon Kim received his BSEE degree from Kyungpook National University, Daegu, Korea, in 1993, and the MS and PhD degrees in electronic and electrical engineering from Pohang University of Science and Technology (POSTECH), Pohang, Korea, in 1995 and 1999, respectively. From July 2003 to June 2004, he studied at the COSY group at the Ruhr-University of Bochum, Germany. He was a senior member of technical staff at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He is currently working as an assistant professor with the Department of Computer Engineering, Pusan National University, Busan, Korea. His research interests include RFID technology, sensor networks, information security, and computer architecture. Currently, his main research focus is on mobile RFID technology and sensor networks, public key cryptosystems and their security issues. He is a member of the IEEE, IEEE Computer Society, and IACR.