PAPER Special Section on Information and Communication System Security

FPGA Implementation of Various Elliptic Curve Pairings over Odd Characteristic Field with Non Supersingular Curves

Yasuyuki NOGAMI[†a)], **Hiroto KAGOTANI**[†], **Kengo IOKIBE**[†], ***Members***, **Hiroyuki MIYATAKE**[††], ***and*** **Takashi NARITA**[††], ***Nonmembers***

SUMMARY Pairing-based cryptography has realized a lot of innovative cryptographic applications such as attribute-based cryptography and semi homomorphic encryption. Pairing is a bilinear map constructed on a torsion group structure that is defined on a special class of elliptic curves, namely pairing-friendly curve. Pairing-friendly curves are roughly classified into supersingular and non supersingular curves. In these years, non supersingular pairing-friendly curves have been focused on from a security reason. Although non supersingular pairing-friendly curves have an ability to bridge various security levels with various parameter settings, most of software and hardware implementations tightly restrict them to achieve calculation efficiencies and avoid implementation difficulties. This paper shows an FPGA implementation that supports various parameter settings of pairings on non supersingular pairing-friendly curves for which Montgomery reduction, cyclic vector multiplication algorithm, projective coordinates, and Tate pairing have been combinatorially applied. Then, some experimental results with resource usages are shown.

key words: *elliptic curve cryptography, pairing-based cryptography, odd characteristic, FPGA implementation*

1. Introduction

Pairing-based cryptography has attracted many researchers since it realizes a lot of innovative cryptographic applications such as ID-based cryptography [1], time-release encryption [2], anonymous authentication [3], attribute-based cryptography [4], and semi homomorphic encryption [5]. There is no need to say that it should be implemented on various secure applications and devices for satisfying many complicated security demands in the coming IoT (Internet of Things) era. Such activities on pairing are based on sufficient security evaluations [6]–[8] and a bilinearity between rational point groups on elliptic curve and a multiplicative group in a certain finite field. Therefore, the security of pairing is based on not only elliptic curve discrete logarithm problem (ECDLP) but also discrete logarithm problem (DLP). In order to bridge these problems efficiently, so-called embedding degree plays an important role. In practice, it is chosen from 2 to 20 for balancing the problems.

Pairing requires a special curve that has a torsion group structure and is usually called pairing-friendly curve. Pairing-friendly curves are roughly classified into two types: one is supersingular curve and the other is non supersingular curve. It is known that curves in the former class have less embedding degrees than those in the latter one. Therefore, sometimes the former class leads to insecure parameter settings as reported in Hayashi's report [6]. On the other hand, elliptic curves in the latter class supports various kinds of embedding degrees, therefore they are particularly focused on in these years. Especially, Barreto-Naehrig (BN) curve is recently focused on because its embedding degree is 12. In detail, 256-bit ECDLP and 3072-bit DLP is efficiently bridged by the embedding degree 12 of BN curve. Thus, there are a lot of reports on efficient software and hardware implementations of pairing on BN curve [9]–[14].

The best parameter setting around the embedding degrees and the sizes of ECDLP and DLP will be changed according to the situations and environments such as device resources, security levels, and so on. Thus, the researchers and developers want to simulate pairing-based cryptographies with various parameter settings. As an efficient programming library that supports pairings on several kinds of pairing-friendly curve*, Pairing-Based Cryptography (PBC) [15] has been provided. Alternatively, this work provides an environment for simulating pairings on *various* kinds of pairing-friendly curve on FPGA. Since it is difficult to achieve both calculation efficiency and scalability for parameter settings on FPGA, the first priority of this work is to achieve a wide scalability for parameter settings.

In order to realize the scalability for parameter settings together with a better computational efficiency, this work has applied several mathematical and algorithmic tools such as Montgomery reduction [16] for representing and efficiently calculating multi-precision arithmetic, cyclic vector multiplication algorithm (CVMA) [17] for vector multiplications in extension field, projective coordinates for representing rational points and efficiently calculating elliptic curve additions, Montgomery powering ladder [7] for securely carrying out scalar multiplications of rational point, Tate pairing for various kinds of pairing-friendly curve, and so on. Among them, CVMA and Tate pairing play a key role for realizing the scalability, and the other techniques contribute to the computational efficiency. As the target FPGA board, this work has used SAKURA-X [18] that belongs to the series of SASEBO (Side-channel Attack Stan-

Manuscript received June 3, 2015.
Manuscript revised September 28, 2015.
Manuscript publicized January 13, 2016.

[†]The authors are with Graduate School of Natural Science and Technology, Okayama University, Okayama-shi, 700–8530 Japan.
[††]The authors are with Tokyo Electron Device Limited, Yokohama-shi, 221–0056 Japan.
a) E-mail: yasuyuki.nogami@okayama-u.ac.jp

DOI: 10.1587/transinf.2015ICP0018

*As easily understood, the best calculation efficiency will be achieved by restricting the target pairing-friendly curves.

Copyright © 2016 The Institute of Electronics, Information and Communication Engineers

dard Evaluation BOard) [19]. Then, this paper shows some experimental results on vector multiplication, scalar multiplication, and pairing together with their resource usages. In order to discuss the efficiency of our implementation, some previous works are briefly introduced; however, it basically becomes an unfair comparison because every previous implementation [13], [20] has been optimized for BN curve. In other words, our work will be the first for supporting various kinds of pairings on FPGA.

As a future work, this paper briefly discusses the viewpoint of optimal (twisted) Ate pairing that is one of the most efficient bilinear elliptic curve pairings.

Throughout this paper, p denotes an odd prime number. Then, $\mathbb{F}_{p^m}$ denotes its m-th extension field and $\mathbb{F}^*_{p^m}$ denotes the multiplicative group of $\mathbb{F}_{p^m}$, that is $\mathbb{F}^*_{p^m} = \mathbb{F}_{p^m} - \{0\}$.

2. Preliminaries

This section briefly introduces prime and extension fields, multi-precision arithmetic, elliptic curve cryptography, and pairing-based cryptography.

2.1 Prime Field and Characteristic Size

Pairing-based cryptography such as with Barreto-Naehrig (BN) curve [21] requires arithmetic operations not only in prime field $\mathbb{F}_p$ but also in extension field $\mathbb{F}_{p^m}$. There are four fundamental arithmetic operations as addition, subtraction, multiplication, and division. A division is generally implemented as a multiplication by the inverse element of the divisor. Among the four arithmetic operations a division is the most time-consuming operation. In case of cryptography, these arithmetic operations should be efficiently carried out with multi-precision arithmetic operations. This work has applied Montgomery reduction [16] and inversion [25]. As a related technique, Montgomery trick [22] is often applied for decreasing the number of simultaneous inversions.

In case of pairing-based cryptography defined with elliptic curve cryptography, the securities of elliptic curve discrete logarithm problem (ECDLP) on elliptic curve and discrete logarithm problem (DLP) in extension field need to be both guaranteed efficiently. The embedding degree k introduced in the following section plays a role of bridging the securities. In recent years, the size[†] of ECDLP needs to be more than 160 bits and that of DLP does more than 2048 bits. Thus, from an efficiency viewpoint, the embedding degree 12 that is provided by BN curve is currently the most efficient bridge; however, the best parameter setting around the embedding degrees and the sizes of ECDLP and DLP will be changed according to the situations end environments such as device resources and security levels. Actually, the embedding degrees are selected from 2 to 20.

[†]According to Hasse's theorem [7], it becomes equal to the size of characteristic p of the prime field $\mathbb{F}_p$.

2.2 Extension Field

The construction of extension field $\mathbb{F}_{p^m}$ over prime field $\mathbb{F}_p$ basically requires an irreducible polynomial $f(x)$ of degree m. Extension field $\mathbb{F}_{p^m}$ is understood as an m-th vector space over $\mathbb{F}_p$. Let a zero of $f(x)$ be $\omega \in \mathbb{F}_{p^m}$, a polynomial basis $\mathcal{P}$ is defined as follows.

$$\mathcal{P} = \{1, \omega, \omega^2, \cdots, \omega^{m-1}\}. \tag{1}$$

Then, an arbitrary element $A \in \mathbb{F}_{p^m}$ is represented as a linear combination over $\mathbb{F}_p$ as follows.

$$A = \sum_{i=0}^{m-1} a_i \omega^i, \ a_i \in \mathbb{F}_p. \tag{2}$$

As found from the above form, it is said that a polynomial basis is efficient for a vector multiplication. As another well-known basis, a normal basis $\mathcal{N}$ is given as

$$\mathcal{N} = \{\omega, \omega^p, \omega^{p^2}, \cdots, \omega^{p^{m-1}}\} \tag{3}$$

for which ω needs to satisfy a certain condition [23]. In the same of Eq. (2), an arbitrary vector in $\mathbb{F}_{p^m}$ is represented as a linear combination of basis elements in $\mathcal{N}$ over $\mathbb{F}_p$. It is said that normal basis is efficient for Frobenius mapping: $A \to A^p$, that is just p-th power operation.

An inversion A^{-1} for a non-zero vector $A \in \mathbb{F}^*_{p^m}$ is generally calculated by Ito-Tsujii algorithm [24] as follows.

$$A^{-1} = (A \cdot B)^{-1} B, \text{ where } B = \prod_{i=1}^{m-1} A^{p^i}. \tag{4}$$

It is important that $A \cdot B$ in the above equation becomes the product of all conjugates A^{p^i}, $0 \le i < m$. Thus, it becomes a non-zero element in $\mathbb{F}_p$ and therefore the above inversion is calculated in $\mathbb{F}_p$. In the same of prime field arithmetic operations, since a division is performed by a multiplication with the inverse element of the divisor, division is the most time-consuming among the four fundamental arithmetic operations in extension field.

In order to construct extension field arithmetic operations, as described above, an irreducible polynomial that is used for the modular polynomial is generally required. When various extension fields should be examined as the motivation of this research, preparing irreducible polynomials of various degrees and/or implementing calculation algorithms optimized for each irreducible polynomial are often inconvenient for researchers. In order to overcome this inconvenience, this work applies cyclic vector multiplication algorithm (CVMA) [17].

2.3 Multi-Precision Arithmetic Operations

This section introduces Montgomery representation and reduction in order to efficiently calculate multi-precision arithmetic operations in this work.

Algorithm 1: Montgomery reduction

Input: $S, N, M, -M^{-1}$.
Output: Redc(S).
1 $t \leftarrow (S + (S(-M^{-1}) \bmod N)M)/N$.
2 **if** $t \geq M$ **then**
3 $\quad t \leftarrow t - M$.
4 return t

Montgomery reduction is an algorithm that allows multi-precision modular multiplication to be performed efficiently [16] (*see* **Algorithm** 1). The Montgomery reduction of a non-negative integer S is defined as follows, where $0 < M < N$, $0 \leq S < MN$, $\gcd(M, N) = 1$, $N(N^{-1}) \equiv 1 \pmod{M}$:

$$\mathrm{Redc}(S) = SN^{-1} \bmod M. \tag{5}$$

Montgomery showed the function could be calculated efficiently using $-M^{-1}$ (where $M(-M^{-1}) \equiv -1 \pmod{N}$) calculated in advance. Since the characteristic in our research is an odd prime number, we can choose N as a 2's power with $M = p$. In this case, the Montgomery reduction algorithm no longer needs to perform modulo and division by N; they are replaced by bit operations. This reduces computation time especially for multi-precision modular multiplication. In order to use Montgomery reduction for modular multiplication, we convert integers into so called Montgomery representation by multiplying N with modulo M in advance. The Montgomery representation X of an integer x can also be calculated by Montgomery reduction $X = xN \bmod M = \mathrm{Redc}(x(N^2 \bmod M))$, where $N^2 \bmod M$ can be calculated in advance. The Montgomery reduction of the product of two Montgomery representations $Z = \mathrm{Redc}(XY)$ can be converted into a normal integer as $z = Z(N^{-1}) \bmod M = \mathrm{Redc}(Z)$ and z is easily shown to be equal to $xy \bmod M$.

Montgomery also showed addition and subtraction in Montgomery representation were the same as those in normal representation. It means a conversion to/from Montgomery representation has to be performed only once for a batch of the complicated computation. The authors have referred to Lórencz's work [25] for Montgomery inversion.

2.4 Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) whose characteristic p of the definition field is larger than 3 is generally constructed over the following elliptic curve E.

$$E : y^2 = x^3 + ax + b,\ a, b \in \mathbb{F}_p. \tag{6}$$

The solutions of E are called rational points. When the rational points on the curve E are defined over $\mathbb{F}_{p^m}$, the set of rational points including the infinity point O forms an additive Abelian group and the group is denoted by $E(\mathbb{F}_{p^m})$.

Let $R(x_R, y_R)$ and $Q(x_Q, y_Q)$ be rational points on the curve, the addition for rational points is defined as follows.

$$T(x_T, y_T) = R(x_R, y_R) + Q(x_Q, y_Q). \tag{7}$$

$$\lambda = \begin{cases} (3x_R^2 + a)/2y_R & \text{if } R = Q \text{ and } y_R \neq 0 \\ (y_R - y_Q)/(x_R - x_Q) & \text{else if } R \neq \pm Q \\ \phi & \text{otherwise} \end{cases}, \tag{8a}$$

$$\begin{pmatrix} x_T \\ y_T \end{pmatrix} = \begin{cases} \begin{pmatrix} \lambda^2 - x_R - x_Q \\ \lambda(x_R - x_T) - y_R \end{pmatrix} & \text{if } \lambda \neq \phi \\ O & \text{otherwise} \end{cases}. \tag{8b}$$

Corresponding to the cases of $R = Q$ or $R \neq Q$, it is generally called elliptic curve addition (ECA) or elliptic curve doubling (ECD), respectively. As shown in the calculations, an inversion is required. In order to avoid the inversions, this work applies projective coordinates [7].

Then, a scalar multiplication $[s]R$ with a scalar s is considered as follows.

$$[s]R = \sum_{i=0}^{s-1} R. \tag{9}$$

In case of ECC, the order of $E(\mathbb{F}_{p^m})$ becomes as large as 160 bits, correspondingly the scalar s also becomes the same bit length. In order to efficiently calculate a scalar multiplication Eq. (9), the well known binary method [7] is available; however, from the security viewpoints on side channel attacks, it is not recommended to apply the method as it is. Thus, this work applies Montgomery powering ladder technique for calculating a scalar multiplication.

In what follows, the smallest number r such that $[r]R = O$ is called the order of rational point R and $E(\mathbb{F}_{p^m})[r]$ denotes the subgroup in $E(\mathbb{F}_{p^m})$ that consists of all rational points of order r including the infinity O.

2.5 Pairing-Based Cryptography

As previously introduced, the bilinearity of pairing has contributed to many kinds of recent innovative cryptographic applications. A pairing is defined as a two to one mapping from two rational point groups on a certain elliptic curve E to a multiplicative group in a certain extension field $\mathbb{F}_{p^k}$. The extension degree k of the minimal extension field such that the bilinear mapping is available is called embedding degree. Then, the bilinear mapping e is defined as

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{F}_{p^k}^*/(\mathbb{F}_{p^k}^*)^r, \tag{10}$$

where $\mathbb{G}_1$, $\mathbb{G}_2$ are cyclic groups of rational points of order r in $E(\mathbb{F}_{p^k})$ and the residue group $\mathbb{F}_{p^k}^*/(\mathbb{F}_{p^k}^*)^r$ is congruent to the multiplicative subgroup of order r in $\mathbb{F}_{p^k}^*$.

As an important restriction found from Eq. (10), the elliptic curve on which pairing is defined needs to have a torsion group structure such as $\mathbb{Z}_r \oplus \mathbb{Z}_r$ of rank 2[†]. Thus, elliptic curves that has a group structure of rank 2 are called pairing-friendly curve and then the embedding degree k for order r

[†]It is known that the group structure of elliptic curve defined over finite field has a group structure of rank 1 or 2.

is decided by the minimal number such that r divides $p^k - 1$. As one of classes of pairing-friendly curves, supersingular curves whose embedding degrees are less than or equal to 6 are known. The embedding degrees of supersingular curves are relatively small from those of non supersingular pairing-friendly curves such as BN curve.

Since the smallness sometimes affects some insecurity [6], non supersingular pairing-friendly curves have attracted many researchers from the viewpoints of not only cryptographic protocols but also efficient implementations on both software [21] and hardware [21]. Currently BN curve seems to be the most attractive because its embedding degree 12 bridges the security gaps between ECDLP and DLP the most efficiently; however, various kinds of non supersingular pairing-friendly curves should be studied and available on both software and hardware. A library for "paring-based cryptography (PBC)" [15] provides a programming library that supports several kinds of pairings; however, for hardware, researchers have only focused on BN curve [13], [20]. It is not easy to realize an efficient implementation that supports many kinds of non supersingular pairings at the same devise.

Pairing-friendly curves are roughly classified into two types: one is supersingular curve and the other is non supersingular curve. Supersingular curve realizes very efficient pairing calculations such as η_T pairing [26]; however, its embedding degree that bridges between ECDLP and DLP is small such as less than or equal to 6. Thus, sometimes it leads to insecure situations [6]. On the other hand, non supersingular pairing-friendly curves, especially BN curve, are able to have larger embedding degree and therefore they are recently focused on from software and hardware viewpoints [13], [15].

In case of non supersingular pairing friendly curves, Weil and Tate pairing are well known [7]. Especially for BN curve, more efficient pairings such as optimal and Xate pairings have been proposed [10], [11]. As found in these efficient pairings, restricting curves and parameters leads to more efficient but complicated implementations. On the other hand, Weil and Tate pairings[†] does not restrict pairing-friendly curves. Since the main purpose of this paper is to provide a simulation environment on FPGA with supporting various kinds of pairings with various non supersingular pairing-based curves, thus this paper implements Tate pairing with efficient algorithms on FPGA.

2.5.1 Pairing-Based Cryptographic Protocols

Recent innovative cryptographic protocols such as time release encryption [2] are attribute-based authentication [4] are realized by the bilinearity of pairing. Security protocols basically request the parameter settings for pairing such as characteristic p and the order of elliptic curve. In order to support various kinds of pairing-based protocols, various kinds of parameter settings of pairing should be supported on both software and hardware. It is one of important motivations of this research to provide a simulation environment for pairing-based protocols on FPGA.

[†]Weil pairing is obtained by twice Tate pairing calculations.

3. Efficient Algorithmic Techniques for Various Pairings

This section introduces efficient algorithmic techniques such as cyclic vector multiplication algorithm (CVMA), projective coordinates, and Tate pairing for realizing various pairings on FPGA. Especially, CVMA plays an important role to support various parameter settings.

3.1 Cyclic Vector Multiplication Algorithm

As previously introduced, an irreducible polynomial of degree m over $\mathbb{F}_p$ is necessary for constructing arithmetic operations in extension field $\mathbb{F}_{p^m}$. Its form such as coefficients, for example, usually affects the efficiency and complexity of calculating and implementing the arithmetic operations. Thus, supporting various kinds of extension fields required for various pairings is not a simple problem.

Cyclic vector multiplication algorithm (CVMA) [17] has been proposed to be able to overcome the above inconvenience. CVMA has two typical features as follows. First, CVMA does not need irreducible polynomials explicitly because the idea of CVMA is just based on cyclotomic polynomial. Thus, a simple check for parameter settings is only required. Second, a vector multiplication by CVMA only iterates additions and multiplications by a simple routine. Such a calculation structure is mostly suitable for hardware implementations.

3.1.1 Parameter Settings for CVMA

When one applies CVMA for a multiplication of vectors in $\mathbb{F}_{p^m}$, the following conditions together with an additional parameter h need to be satisfied. It is just the existence condition of Gauss period normal basis (GNB) in $\mathbb{F}_{p^m}$ [23].

Condition 1 *Let h be a positive integer such that $q = hm+1$ becomes a prime number and let θ be the order of p modulo q. Then, Gauss period normal basis of period r exists in $\mathbb{F}_{p^m}$ if and only if* $\gcd(\theta, m) = 1$. □

It is noted that, according to our previous work [17], all of non supersingular pairing-friendly curves for cryptography are able to be defined over extension fields that has Gauss period normal basis[††]. Let $\{\gamma, \gamma^p, \cdots, \gamma^{p^{m-1}}\}$ be the Gauss period normal basis with a certain h in $\mathbb{F}_{p^m}$, CVMA calculates a vector multiplication as shown in **Algorithm** 2. It is found that CVMA works without knowing the detail of the GNB. As found from the algorithm, the parameter h is preferred to be small for the calculation efficiency. In practice, as also shown in our previous work [27], h is mostly less than 10 and thus the smallest h is easily found.

[††]In brief, CVMA just requires that $p > m$. It is satisfied for every non supersingular pairing-friendly curves for cryptography.

Algorithm 2: Cyclic vector multiplication algorithm

Input: $X = \sum_{i=0}^{m-1} x_i \gamma^{p^i}$, $Y = \sum_{i=0}^{m-1} y_i \gamma^{p^i}$, $x_i, y_i \in \mathbb{F}_p$.

Output: $Z = XY = \sum_{i=0}^{m-1} z_i \gamma^{p^i}$, $z_i \in \mathbb{F}_p$.

Preparation steps: // Preparation of the calculation table.

1 Prepare a primitive h-th root d of unity in $\mathbb{F}_r$.
2 $\epsilon[0] \leftarrow m$.
3 **for** $i = 0$ **to** $m - 1$ **do**
4 **for** $j = 0$ **to** $h - 1$ **do**
5 $\epsilon[(p^i d^j \bmod r)] \leftarrow i$.
6 **for** $i = 0$ **to** $m - 2$ **do**
7 **for** $j = i + 1$ **to** $m - 1$ **do**
8 **for** $l = 0$ **to** $k - 1$ **do**
9 $\beta[i][j][l] \leftarrow \epsilon[(p^i + p^j d^l \bmod r)]$.

Main steps:

10 **for** $i = 0$ **to** m **do**
  $v[i] \leftarrow 0$.
11 **for** $i = 0$ **to** $m - 2$ **do**
12 **for** $j = i + 1$ **to** $m - 1$ **do**
13 $u \leftarrow (x_i - x_j)(y_i - y_j)$. // Montgomery reduction
14 **for** $l = 0$ **to** $k - 1$ **do**
15 $v[\beta[i][j][l]] \leftarrow v[\beta[i][j][l]] + u$. // non mod p
16 **if** *h is odd* **then**
17 $w \leftarrow hv[m]$. // non mod p
18 **for** $i = 0$ **to** $m - 1$ **do**
19 $z_i \leftarrow w - x_i y_i - v[i] \bmod p$.
20 **else**
21 **for** $i = 0$ **to** $m - 1$ **do**
22 $z_i \leftarrow v[i] - x_i y_i \bmod p$.

The most remarkable feature of CVMA is that, as found from the algorithm, it supports an arbitrary pair of p and m since it does not need any irreducible polynomials as the modular polynomial. In addition, it just iterates additions/subtractions and multiplications. The above features are both suitable for hardware implementations. In brief, from the viewpoint of calculation cost, CVMA needs $m(m+1)/2$ multiplications at line 13 that is implemented by Montgomery reduction technique in this work.

The modulo p operation after an addition of two integers as elements in $\mathbb{F}_p$ is generally implemented by a subtraction by p; however, this work lazily carries out such a modulo p operation as shown in lines 15, 19, and 22.

3.2 Affine and Projective Coordinates

In case of pairing-based protocols, since a pairing calculation is the most time-consuming one among the others, scalar multiplications defined by Eq. (9) are possibly used so as to minimize the number of required pairing calculations. A scalar multiplication iterates elliptic curve additions (doublings) and an elliptic addition consists of the four fundamental arithmetic operations in the base field including an inversion as shown in Eqs. (8).

Then, since an inversion is the most time-consuming among the four fundamental arithmetic operations in the base field, projective coordinate[†] is often applied especially for hardware implementation. As introduced below, an elliptic curve addition with projective coordinates does not require any inversions. Then, elliptic curve addition $T = R+Q$ and doubling $[2]R = R + R$ with projective coordinates on $E : y^2z = x^3 + axz^2 + bz^3$ are given as follows.

$$T(x_T, y_T, z_T) = R(x_R, y_R, z_R) + Q(x_Q, y_Q, z_Q). \tag{11}$$

$$\begin{pmatrix} x_T \\ y_T \\ z_T \end{pmatrix} = \begin{pmatrix} \lambda_2\lambda_3 \\ \lambda_1(\lambda_2^2 x_R z_Q - \lambda_3) - \lambda_2^3 y_R z_Q \\ \lambda_2^3 z_R z_Q \end{pmatrix}. \tag{12a}$$

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} = \begin{pmatrix} y_Q z_R - y_Q z_R \\ x_Q z_R - z_Q x_R \\ \lambda_1^2 z_R z_Q - \lambda_2^3 - 2\lambda_2^2 x_R z_Q \end{pmatrix}. \tag{12b}$$

$$\begin{pmatrix} x_{[2]R} \\ y_{[2]R} \\ z_{[2]R} \end{pmatrix} = \begin{pmatrix} \lambda_5\lambda_7 \\ \lambda_4(2\lambda_6 - \lambda_7) - 2(\lambda_5 y_R)^2 \\ \lambda_5^3 \end{pmatrix}. \tag{13a}$$

$$\begin{pmatrix} \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \end{pmatrix} = \begin{pmatrix} az_R^2 + 3x_R^2 \\ 2y_T z_T \\ x_T y_T \lambda_5 \\ \lambda_4^2 - 4\lambda_6 \end{pmatrix}. \tag{13b}$$

Thus, projective coordinates are used throughout the FPGA implementation in this work. In brief, the case that z coordinate is zero corresponds to the infinity point O. In brief, elliptic curve addition and doubling with projective coordinates need 14 and 11 multiplications in the base field, respectively. They do not need any inversions.

3.3 Tate Pairing

Tate pairing [7] is available for every pairing-friendly curve. As previously introduced, pairing requires a torsion group structure of rank 2 defined as follows.

$$E(\mathbb{F}_{p^k})[r] \approx \mathbb{Z}_r \oplus \mathbb{Z}_r, \tag{14}$$

where k and r denote the embedding degree and order of non-zero rational points in the group. Note that, for simplicity, r is a prime number in this paper. In addition, as a property of pairing, the order r divides $p^k - 1$.

Tate pairing $\tau(\cdot, \cdot)$, that is a two to one mapping, maps two points $R, Q \in E(\mathbb{F}_{p^k})[r]$ to a vector $A \in \mathbb{F}_{p^k}$ as follows.

$$\tau(R, Q) = f_{r,R}(Q)^{(p^k-1)/r}. \tag{15}$$

In general, R and Q are rational points in $\mathbb{G}_1$ an $\mathbb{G}_2$ respectively, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are certain cyclic groups of rational points of order r in $E(\mathbb{F}_{p^k})[r]$. In practice, $\mathbb{G}_1$ is set by $E(\mathbb{F}_p)[r]$ for efficiency since the x and y coordinates of every rational point $R \in \mathbb{G}_1$ are elements in prime field $\mathbb{F}_p$. On the other hand, $\mathbb{G}_2$ is mostly defined over the embedded extension field $\mathbb{F}_{p^k}$. The bilinearity that enables recent innovative

[†]Equations (8) are the definition by affine coordinates.

pairing-based applications is represented as follows.

$$\tau([a]R,[b]Q) = \left(f_{r,R}(Q)^{(p^k-1)/r}\right)^{ab} = \tau([a]R,[b]Q)^{ab}. \tag{16}$$

Tate pairing defined by Eq. (15) has two calculation parts. First, the exponentiation by $(p^k-1)/r$ is called final exponentiation. This work supposes to apply the well known binary method for a part of the final exponentiation [12]. Binary method has been widely used for not only exponentiations but also scalar multiplications because it does not restrict its applicable parameters. Then, $f_{r,R}(Q)$ is actually calculated by Miller's algorithm [28]. It has a similar structure of the binary method as shown in **Algorithm** 3. It efficiently uses the line evaluations Eqs. (17), (18), and (19), where S in the equations denotes $T+P$ or $[2]T$.

$$Ln_{T,R}(Q) = \begin{cases} 1 & \text{if } T=O \text{ or } R=O \text{ or } Q=O \\ x_Q z_T - z_T & \text{else if } \lambda_2 = 0 \\ \lambda_1(x_Q z_T - x_T) - \lambda_2(y_Q z_T - y_T) & \text{otherwise} \end{cases}. \tag{17a}$$

$$Ld_{T,R}(Q) = \begin{cases} 1 & \text{if } T=O \text{ or } R=O \text{ or } Q=O \\ z_T & \text{else if } \lambda_2 = 0 \\ \lambda_2 z_T & \text{otherwise} \end{cases}. \tag{17b}$$

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix} = \begin{pmatrix} y_R z_T - y_T \\ x_R z_T - z_T \end{pmatrix}. \tag{17c}$$

$$Ln_{T,T}(Q) = \begin{cases} 1 & \text{if } T=O \text{ or } Q=O \\ \lambda_4(x_Q z_T - x_T) - \lambda_5(y_Q z_T - y_T) & \text{otherwise} \end{cases}. \tag{18a}$$

$$Ld_{T,T}(Q) = \begin{cases} 1 & \text{if } T=O \text{ or } Q=O \\ \lambda_5 z_T & \text{otherwise} \end{cases}. \tag{18b}$$

$$\begin{pmatrix} \lambda_4 \\ \lambda_5 \end{pmatrix} = \begin{pmatrix} a z_T^2 + 3x_T^2 \\ 2y_T z_T \end{pmatrix}. \tag{18c}$$

$$Vn_S(Q) = \begin{cases} 1 & \text{if } S=O \text{ or } Q=O \\ x_Q z_S - x_S & \text{otherwise} \end{cases}. \tag{19a}$$

$$Vd_S(Q) = \begin{cases} 1 & \text{if } S=O \text{ or } Q=O \\ z_S & \text{otherwise} \end{cases}. \tag{19b}$$

As shown in **Algorithm** 3, it is based on the representation of rational points with projective coordinates. Thus, basically no inversions in the base field are required; however, it is noted that, as shown in the algorithm, Tate pairing requires only one inversion at the last step for which this work has implemented Montgomery inversion [25]. The cost evaluation of Miller's algorithm is not simple because this work does not fix the target parameters. It is noted that, as shown in the algorithm, Miller's algorithm consists of elliptic curve additions and vector multiplications.

Algorithm 3: Miller's algorithm of Tate pairing

Input: $R \in \mathbb{G}_1,\ Q \in \mathbb{G}_2$.
Output: $f_{r,R}(Q)$.
$u \leftarrow 1, v \leftarrow 1, T \leftarrow R$.
for $i = \lfloor \log_2 r \rfloor$ **downto** 1 **do**
  $f \leftarrow f^2 \cdot Ln_{T,T}(Q) \cdot Vd_{[2]T}(Q)$.
  $h \leftarrow h^2 \cdot Vn_{[2]T}(Q) \cdot Ld_{T,T}(Q)$.
  $T \leftarrow [2]T$.
  if $r_i = 1$ **then**
    $f \leftarrow f \cdot Ln_{T,R}(Q) \cdot Vd_{T+R}(Q)$.
    $h \leftarrow h \cdot Vn_{T+R}(Q) \cdot Ld_{T,R}(Q)$.
    $T \leftarrow T + R$.
$f \leftarrow fh^{-1}$
return f
* r_i denotes the i-th bit of r.

3.4 Binary Method and Montgomery Powering Ladder

As previously described, binary method has been used for various situations of exponentiation and scalar multiplication. Binary method iterates squarings and multiplications corresponding to the bit information of the exponent and therefor it is recently said that binary method especially for hardware implementation is not recommended from the viewpoint of side channel attacks. Of course, if the exponent is a public information such as the final exponentiation of pairing, there is no problem to apply binary method and thus a lot of improvements of final exponentiations, especially for the cases with BN curve, have been proposed based on binary method [12]. Moreover, it is known that a vector exponentiation and also scalar multiplication defined over extension field are drastically improved by using Frobenius mapping [29]; however, this paper simply applies binary method for the final exponentiation.

On the other hand, scalar multiplications for pairing-based cryptography are mostly operated with secret scalars. Thus, in the same of RSA cryptography, it is said that binary method is not recommended for scalar multiplications from the viewpoint of side channel attacks. Thus, this work has also considered Montgomery powering ladder (MPL) [7] as one of countermeasures for side channel attacks. The detail of MPL implementation is shown in **Appendix** A. It should be noted that this paper mainly deals with pairing implementation.

4. FPGA Implementation and Experimental Results

This section introduces our implementation of pairing on FPGA and shows some experimental results. After that, this paper briefly discusses the viewpoint of optimal (twisted) Ate pairing that is one of the most efficient bilinear elliptic curve pairings.

4.1 Target FPGA Board: SAKURA-X

Figure 1 is the appearance of the target board SAKURA-X [18] that belongs to the series of SASEBO (Side-channel Attack Standard Evaluation BOard) [19]. Thus, it is arranged for attacking side channel information during cryptographic calculations on the boards. On the target board SAKURA-X, this work has implemented Tate pairing available for various parameter sets together with CVMA, where their theoretic parts has been introduced in the previous sections. Thus, one of our future works will be attacking its

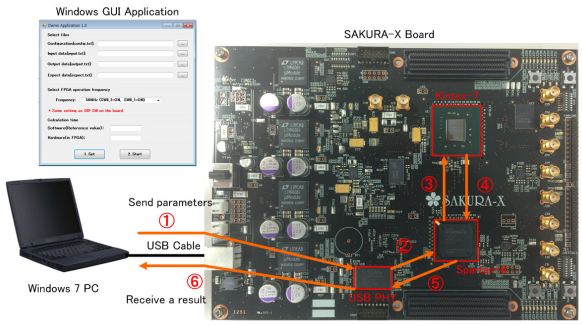


Fig. 1 Appearance of SAKURA-X simulation board.

Table 1 Environment for development.

Item	Description
OS	Windows 7
Simulation tool	ModelSim SE 6.6D
Synthesize tool	ISE Foundation (ISE Service Pack 14.6)
Development language	Verilog HDL
Board	SAKURA-X (SASEBO-GIII) [18]
FPGAs on board	Kintex-7 (XC7K160T) Spartan-6 (XC6SLX45)
Compiler/Library	gcc(GCC) 4.8.3, lgmp
GUI Application development tool	Microsoft Visual Studio Professional 2013

Table 2 Operation frequency.

Operation	Frequency [MHz]
LBUS I/F Clock	24
CVMA	50
ECC (*see* **Appendix** A)	50
Miller's algorithm	50

side channel information during pairing calculations.

This paper mainly shows the implementation of Tate pairing; however, as described in this paper, CVMA, elliptic curve addition by projective coordinates, and scalar multiplication by Montgomery powering ladder (*see* **Appendix** A) were also implemented on SAKURA-X.

Tables 1 and 2 respectively show the computational environment and operation frequencies. Table 3 also shows the computational resources of Kintex-7. Then, CVMA and Tate pairing are implemented on Kintex-7.

4.2 Implementation on SAKURA-X

This section briefly introduces parameters and initializations of our implementation.

4.2.1 Parameters and Initializations

Our implementation needs the following parameters and initializations. They are firstly inputted and performed.

Parameters:
[CVMA]

- characteristic p and embedding (extension) degree k.

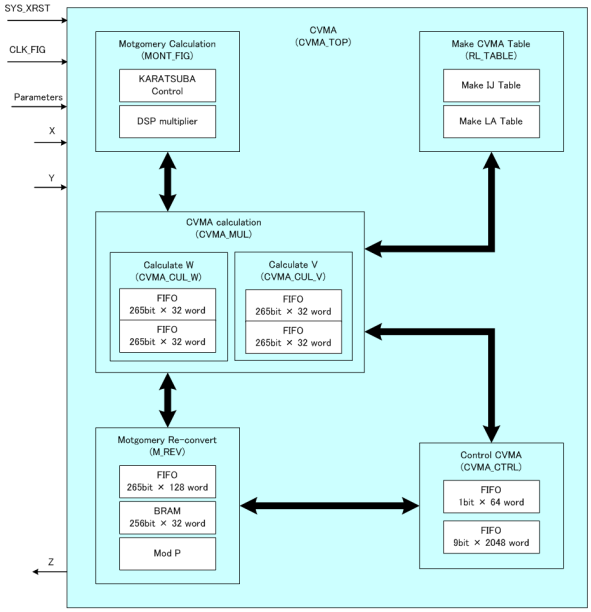


Fig. 2 Block image of CVMA.

- parameter h such that **Condition** 1 is satisfied.

[Montgomery reduction]

- N as a power of 2, this paper sets $N = 2^{78}$.
- $M = p$, $M^{-1} \bmod N$ (actually $p^{-1} \bmod 2^{78}$).

Initializations:

- Calculation table of CVMA (preparation steps, **Algorithm** 2).

4.2.2 CVMA and Vector Exponentiation

Figure 2 shows the design of CVMA block that basically consists of calculating a table (CVMA Table), iterating additions/subtractions/multiplications, and Montgomery reduction for multiplications. In detail, the iterations are carried out according to the schedule stored in the table and each multiplication in the iterations is carried out by Montgomery reduction. In this work, CVMA is also used for the final exponentiation of Tate pairing, where the exponent is a fixed number as shown in Eq. (15).

Then, Table 3 shows the computational resources required for the CVMA block. As found in Fig. 2, it also includes Montgomery reduction block and thus the resources for Montgomery reduction are also shown in the table.

4.2.3 Miller's Algorithm for Pairing

Figure 3 shows the design of Miller's algorithm that consists of iterating vector additions/subtractions/multiplications, Montgomery inversion, and CVMA block. Miller's algorithm **Algorithm** 3 is written with elliptic curve additions and doublings; however, they are broken down

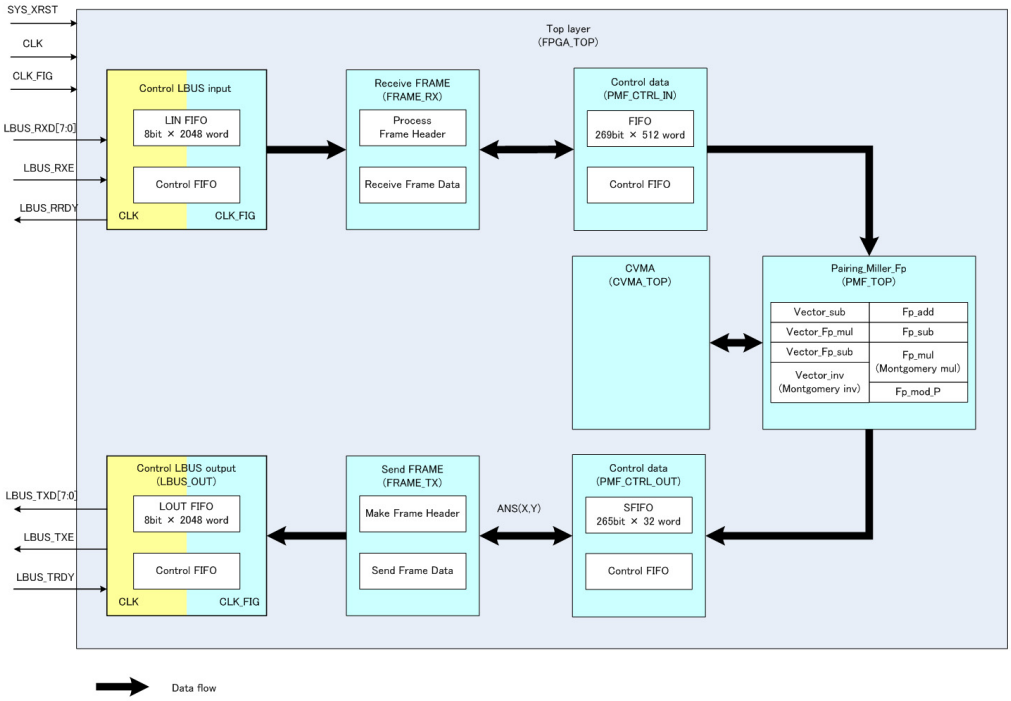


Fig. 3 Block image of Miller's algorithm.

Table 3 Number of circuit blocks used in Kintex-7.

Circuit block	Resource	Usage			
		CVMA		Miller's algorithm (including CVMA)	
Slice LUTs	101,400	12,911 (5,709)	13%	68,301	67%
Slice Registers	202,800	9,114 (4,088)	4%	42,129	21%
Occupied Slices	25,350	4,305 (1,846)	17%	20,438	81%
RAMB36+RAMB18	325	33 (0)	3%	153	16%
BUFG (clock buffer)	650	0 (0)	0%	2	6%
DSP48E	32	143 (45)	24%	233	39%

Remark: The data in ( ) shows the resources used for Montgomery reduction.
The detail of circuit blocks are introduced in the manual of Kintex-7 [30].

into more fundamental arithmetic such as vector additions/subtractions and multiplications when it is implemented on FPGA. Vector multiplications in the iteration are carried out by CVMA and finally a vector division (a multiplication after an inversion, *see* line 10 in **Algorithm** 3) is carried out.

Then, Table 3 shows the computational resources required for the Miller's algorithm implementation. It is noted that a part of the final exponentiation is carried out by binary method with CVMA (*see* [12]).

4.3 Experimental Results

This section especially observes the calculation times of an exponentiation on CVMA and Miller's algorithm of Tate pairing with various parameter settings. It is the most significant contribution of this work that pairings for various parameter settings are able to be simulated on the same FPGA board. The reason why this simulation separately deals with Miller's algorithm and the final exponentiation is that, as previously introduced, most of improvements for the final exponentiation are separately discussed [12].

4.3.1 Calculation Time of an Exponentiation

Table 4 shows the calculation time for a vector multiplication with CVMA and the average calculation time for an exponentiation with CVMA, where the exponent for the latter is chosen from 256-bit random numbers. As shown in Eq. (15), the bit length of the exponent basically becomes

$$L = \left\lceil \log_2 \left((p^k - 1)/r \right) \right\rceil. \tag{20}$$

If the target is BN curve of 256-bit characteristic p, the bit length L becomes more than 2800 bits because $k = 2$ and $\log_2 r \approx 256$. Thus, if any other improvements for the final exponentiation are not applied, the target bit length of the binary method becomes the above L. However, as introduced in Sect. 3.4, recent improvements achieve to reduce the actual bit length of the final exponentiation to almost the bit length of characteristic p [12]. Thus, this simulation has concentrated to demonstrate 256-bit exponentiations for each parameter setting.

According to the result, it is found that the parameter h of CVMA slightly affects the calculation time on the same embedding degree k. Thus, in the following simulations, the smallest h is only focused on for each parameter settings of

Table 4 Average calculation times of a multiplication and an 256-bit exponentiation* in $\mathbb{F}_{p^k}$ with CVMA (the size of p is fixed to 256-bit).

k	h	multiplication [μs]	exponentiation [ms]
2	3	1.28	0.86
3	2	2.06	2.17
4	7	3.16	4.49
5	2	4.38	7.87
6	3	5.92	12.67
7	4	7.70	19.54
8	2	9.66	28.32
9	2	11.90	38.79
10	3	14.40	53.27
11	2	17.10	69.74
12	1	20.04	89.07
12	8	20.18	90.24
18	11	43.00	288.82
20	5	52.44	392.65
20	14	52.76	395.93

* every exponent is randomly chosen from 256-bit numbers.

Table 5 Average calculation time of Miller's algorithm for Tate pairing.

k	h	$\lceil \log_2 r \rceil$ [bits]	$k\lceil \log_2 p \rceil$ [bits]	Miller's algorithm [ms]
4	1	255	1020	8.18
5	6	203	1015	12.26
6	2	183	1098	17.62
7	4	166	1162	19.72
8	2	130	1040	21.18
9	2	223	2007	26.08
10	7	202	2020	26.24
11	2	217	2387	36.67
12	1	170	2040	46.63
12	1	250	3000	40.39
12	3	254	3048	40.43
12	6	250	3000	40.77
14	2	83	1162	33.89
14	9	221	3094	44.79
15	4	203	3045	44.15
16	1	163	2608	53.73
18	2	164	2952	56.52

p and k. By the way, from the viewpoint of software implementation, the calculation cost of CVMA has been discussed in our precious work [27].

4.3.2 Calculation Time of Miller's Algorithm

Table 5 shows the calculation time of Miller's algorithm of Tate pairing. When $k = 12$ and 254-bit characteristic p, that is a case of BN curve, Miller's algorithm is carried out for about 40 ms. It is said that it is enough practical together with the parameter scalability of this implementation.

Table 6 shows a comparison of the calculation time of only Miller's algorithm with BN curve of 256-bit order and embedding degree 12. Yao et al.'s work [13] is the latest report of FPGA implementation of optimal ate pairing [10] with BN curve. The most important difference between ours and Yao's is the type of pairing. It is roughly said that Miller's algorithm calculation of optimal ate pairing is four times more efficient than that of Tate pairing. In addition, in the case of BN curve, the calculations at lines 4, 8 and a multiplication by the inverse h^{-1} at line 10 of **Algorithm** 3 does not required because they are canceled by the final exponentiation. This cancellation yields twice more efficiency. Thus, for Ghosh et al.'s work [20], we can give a reasonable comparison with the data on Table 6, where Miller's algorithm calculation of R-ate pairing [9] is also four times more efficient than that of Tate pairing. Although it is difficult to fairly compare them as introduced, it is concluded that our implementation with a wide parameter scalability could have reached to the same efficiency level.

Table 6 Comparison of Miller's algorithm with previous works (BN curve of 254-bit order with embedding degree 12).

Design	$\lfloor \log_2 r \rfloor$	Platform	Freq. [MHz]	Pairing	Delay [ms]
Ours	254	Xilinx (Kintex-7)	50	Tate	40.43
[13]	254	Xilinx (Virtex-6)	210	optimal ate [10]	0.176
[20]	256	Xilinx (Virtex-4)	50	Tate ate R-ate [9]	26.9* 18.8* 12.8*

* estimated by the authors.

4.4 Viewpoint of Optimal Ate Pairing

For some embedding degrees, optimal (twisted) Ate pairing realizes a quite efficient bilinear mapping [10], [13], [31], [32]. The most popular target is of course BN curve whose embedding degree 12 because it is not only efficient but also appropriate for the near future security level.

One of the important contributions of optimal (twisted) Ate pairing is to reduce the number of calculation loops in Miller's algorithm. In the case of BN curve, compared to the case of Tate pairing, it is reduced to 1/4; however, it is not a problem for our implementation. Another one is, that gives a difficult problem for our implementation as it is, to efficiently support proper subfield arithmetic operations. In the case of BN curve for example, optimal (twisted) Ate pairing requires that arithmetic operations not only in $\mathbb{F}_{p^{12}}$ but also in $\mathbb{F}_{p^2}$ are distinguishably operated. Since the purpose of this paper is to support various pairing-friendly curve with Tate pairing, it is not actually optimized for the above requirement. It is one of the most important future challenges of this work.

5. Conclusion

This paper has proposed an FPGA implementation that supports various parameter settings of pairings on non supersingular pairing-friendly curves for which Montgomery reduction, cyclic vector multiplication algorithm, projective coordinates, and Tate pairing have been used. Then, some experimental results with resource usages were shown. Our implementation supports pairing-related operations such as pairing calculation, vector exponentiation, and scalar multiplication for various parameter settings such as characteristic p less than 256 bits and embedding degrees 2 to 20

with non supersingular pairing-friendly curves. Since it is a hardware implementation, its side-channel security should be strictly evaluated as a future work. In addition, a more optimization such as for optimal Ate pairing is also our important future work.

Acknowledgments

This work was supported by Adaptable and Seamless Technology transfer Program (ASTEP), JST.

References

[1] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairings," SCIS2000, pp.26–28, 2000.
[2] K. Chalkias, F. Baldimtsi, D. Hristu-Varsakelis, and G. Stephanides, "Mathematical problems and algorithms for timed-release encryption," Bulletin of the Transilvania University of Brasov, vol.15, no.50, pp.1–4, 2008.
[3] T. Nakanishi and N. Funabiki, "Verifier-local revocation group signature schemes with backward unlinkability from bilinear maps," ASIACRYPT 2005, LNCS 3788, pp.533–548, Springer, 2005.
[4] N. Begum, T. Nakanishi, and N. Funabiki, "Efficient proofs for CNF formulas on attributes in pairing-based anonymous credential system," IEICE Trans. Fundamentals, vol.96-A, no.12, pp.2422–2433, Dec. 2013.
[5] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias, "Semi-homomorphic encryption and multiparty computation," EUROCRYPT'11, LNCS 6632, pp.169–188, Springer, 2011.
[6] T. Hayashi, N. Shinohara, L. Wang, S. Matsuo, M. Shirase, and T. Takagi, "Solving a 676-bit discrete logarithm problem in GF(3^{6n})," IEICE Trans. Fundamentals, vol.95-A, no.1, pp.204–212, Jan. 2012.
[7] H. Cohen and G. Frey, Handbook of Elliptic and Hyperelliptic Curve Cryptography, Chapman & Hall, 2006.
[8] S. Akagi and Y. Nogami, "Exponentiation inversion problem reduced from fixed argument pairing inversion on twistable Ate pairing and its difficulty," IWSEC 2014, LNCS 8639, pp.240–249, Springer, 2014.
[9] E. Lee, H. Lee, and C. Park, "Efficient and generalized pairing computation on Abelien varieties," IACR ePrint archive, available at http://eprint.iacr.org/2008/040.
[10] F. Vercauteren, "Optimal pairings," IEEE Trans. Inf. Theory, vol.56, no.1, pp.455–461, 2010.
[11] Y. Nogami, Y. Sakemi, H. Kato, M. Akane, and Y. Morikawa, "Integer variable χ-based cross twisted Ate pairing and its optimization for Barreto-Naehrig curve," IEICE Trans. Fundamentals, vol.92-A, no.8, pp.1859–1867, Aug. 2009.
[12] M. Scott, N. Benger, M. Charlemagne, L.J.D. Perez, and E.J. Kachisa, "On the final exponentiation for calculating pairings on ordinary elliptic curves," Pairing 2009, LNCS 5671, pp.78–88, Springer, 2009.
[13] G.X. Yao, J. Fan, R.C.C. Cheung, and I. Verbauwhede, "Faster pairing coprocessor architecture," Pairing 2012, LNCS 7708, pp.160–176, Springer, 2013.
[14] TEPLA: University of Tsukuba Elliptic Curve and Pairing Library, http://www.cipher.risk.tsukuba.ac.jp/tepla/index_e.html
[15] PBC: The Pairing-Based Cryptography Library, https://crypto.stanford.edu/pbc/
[16] P.L. Montgomery, "Modular multiplication without trial division," Math. Comput., vol.44, no.170, pp.519–521, 1985.
[17] K. Nekado, Y. Nogami, H. Kato, and Y. Morikawa, "Cyclic vector multiplication algorithm and existence probability of Gauss period normal basis," IEICE Trans. Fundamentals, vol.94-A, no.1, pp.172–179, Jan. 2011.
[18] SAKURA: Hardware security project, http://satoh.cs.uec.ac.jp/SAKURA/index.html
[19] SASEBO: Side-channel Attack Standard Environment BOard, http://www.risec.aist.go.jp/project/sasebo/
[20] S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury, "High speed flexible pairing cryptoprocessor on FPGA platform," Pairing 2010, LNCS 6487, pp.450–466, Springer, 2010.
[21] P.S.L.M. Barreto and M. Naehrig,"Pairing-friendly elliptic curves of prime order," SAC 2005, LNCS 3897, pp.319–331, Springer, 2006.
[22] P.L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," Math. Comput., vol.48, no.177, pp.243–264, 1987.
[23] R. Lidl and H. Niederreiter, Finite Fields, Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 1984.
[24] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases," Information and Computation, vol.78, no.3, pp.171–177, 1988.
[25] R. Lórencz and J. Hlaváč, "Subtraction-free almost Montgomery inverse algorithm," Information Processing Letters, vol.94, no.1, pp.11–14, 2005.
[26] Y. Kawahara, K. Aoki, and T. Takagi, "Faster implementation of η_T pairing over GF(3^m) using minimum number of logical instructions for GF(3)-addition," Pairing 2008, LNCS 5209, pp.282–296, Springer, 2008
[27] H. Kato, Y. Nogami, T. Yoshida, and Y. Morikawa, "Cyclic vector multiplication algorithm based on a special class of Gauss period normal basis," ETRI Journal, vol.29, no.6, pp.769–778, 2007.
[28] V.S. Miller, "Short programs for functions on curves," http://crypto.stanford.edu/miller/miller.pdf
[29] Y. Nogami, Y. Sakemi, T. Okimoto, K. Nekado, M. Akane, and Y. Morikawa, "Scalar multiplication using Frobenius expansion over twisted elliptic curve for Ate pairing based cryptography," IEICE Trans. Fundamentals, vol.92-A, no.1, pp.182–189, Jan. 2009.
[30] http://www.xilinx.com/products/silicon-devices/fpga/kintex-7.html#documentation
[31] X. Zhang, K. Wang, and D. Lin, "On efficient pairings on elliptic curves over extension fields," Pairing 2012, LNCS 7708, pp.1–18, Springer, 2012.
[32] J. Han, Y. Li, Z. Yu, and X. Zeng, "A 65 nm cryptographic processor for high speed pairing computation," IEEE Trans. Very Lagre Scale Integr. (VLSI) Syst., vol.23, no.4, pp.692–701, 2015.

Appendix: Montgomery Powering Ladder

Most of side channel attacks focus on the slight difference between elliptic curve addition and doubling operations in every loop of the iteration. The basic idea of MPL is that both elliptic curve addition and doubling are always carried out in every loop. **Algorithm** 4 shows the algorithm of Montgomery powering ladder (MPL). Therefore, MPL for a scalar multiplication of scalar s requires $\lfloor \log_2 s \rfloor$ elliptic

Algorithm 4: Montgomery powering ladder

Input: $R \in E(\mathbb{F}_{p^k})[r]$, s $(0 \le s < r)$.
Output: $[s]R$.
1 $T_0 \leftarrow O$, $T_1 \leftarrow R$.
2 **for** $i = \lfloor \log_2 s \rfloor$ **downto** 0 **do**
3 **if** $s_i = 1$ **then**
4 $T_0 \leftarrow T_0 + T_1$, $T_1 \leftarrow T_1 + T_1$.
5 **else**
6 $T_1 \leftarrow T_0 + T_1$, $T_0 \leftarrow T_0 + T_0$.
7 return T_0
* s_i denotes the i-th bit of scalar s.

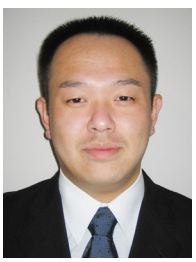
Table A· 1 Average calculation time of a scalar multiplication* in $E(\mathbb{F}_{p^k})$ with Montgomery powering ladder (the size of p is fixed to 256-bit).

k	h	$k\lceil \log_2 p \rceil$	EC addition [ms]	MPL [ms]
1	2	256	0.02	10.19
2	3	512	0.03	15.58
3	2	768	0.04	22.18
4	7	1024	0.06	30.90
6	3	1536	0.10	52.13
12	1	3072	0.30	151.64
18	11	4608	0.60	309.10
20	5	5120	0.73	372.81

* scalars for scalar multiplications are set by 256-bit random numbers.

curve doublings and the same number of additions.

Table A· 1 shows the average calculation time for a scalar multiplication with Montgomery powering ladder, where the scalars are randomly chosen from 256-bit numbers. It is noted that scalar multiplications of our implementation are carried in $E(\mathbb{F}_{p^k})$. Our implementation supports various pair of p and k. When p is a 256-bit prime and $k = 20$ as shown in the table, the security bits becomes more than 5000. It is too strong from the viewpoint of ECC security. Thus, Table A· 1 just demonstrates the scalability of our implementation from which it is observed that the calculation time of course increases as the security bits $k\lceil \log_2 p \rceil$ becomes longer.



Yasuyuki Nogami graduated from Shinshu University in 1994 and received the Ph.D. degree in 1999 from Shinshu University. He is now an associate professor of Okayama University. His main fields of research are finite field theory and its applications such as recent public key cryptographies. He is now studying about elliptic curve cryptography, pairing-based cryptography, Lattice-based cryptography, pseudo random number generator, Advanced Encryption Standard, and homomorphic encryptions. Recently, he is a member of security research group in Okayama university and particularly focusing on IoT security from the viewpoints of software and hardware implementations. He is a member of IEICE and IEEE.

Hiroto Kagotani graduated in 1988 and received the Ph.D. degree in 1994 both from Tokyo Institute of Technology. He is now a senior assistant professor of Okayama University. His research area includes asynchronous logic circuit design, hardware security, and GPGPU applications. He is a member of IEICE.

Kengo Iokibe received B.S. and M.S. degrees in electrical and electronic engineering from Okayama University, Japan in 1997 and 1999, respectively. He also received a Ph.D. degree in electronic engineering from Okayama University in 2005. He is currently an Assistant Professor in the Department of Information and Communication Systems at the Graduate School of Natural Science and Technology, Okayama University, where focuses on security against side-channel analysis attack on cryptographic devices, the design of the power distribution network of integrated circuits for power integrity, signal integrity and EMC, and EMC modeling of power converter circuits. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), the Institute of Electrical and Electronics Engineers (IEEE) and Japan Institute of Electronics Packaging (JIEP).

Hiroyuki Miyatake graduated from Tokyo University of Agriculture and Technology in 2013. He is now an engineer of Inrevium Company (Tokyo Electron Device LTD.).

Takashi Narita graduated from Hokkaido Institute of Technology in 1984. He is now a Senior Engineer Professional of Inrevium Company (Tokyo Electron Device LTD.).