Regular Paper

Search of Elliptic Curves Suitable for Signature

Masaaki Shirase†

†School of Systems Information Science, Future University Hakodate, Japan
shirase@fun.ac.jp

Abstract - Elliptic curve signatures as ECDSA have features that the processing is faster and the signature length is shorter than those of RSA signatures with same security. The use of elliptic curve signatures was decided for the V2X communication with limited bandwidth. However, higher processing speed is required.

In an elliptic curve signature using 256-bit prime p, thousands of modular multiplications $X \cdot Y \bmod p$ performed according to a signature algorithm are dominant. Therefore, how to speed up multiplications and $\bmod p$ computations is one of the objectives of researches on elliptic curve signature implementations. One of speeding up method of reduction $\bmod p$ is to use a special form of prime called pseudo Mersenne prime such that $p = 2^n - k$, where k is a small value. However, in an elliptic curve signature, computation of $\bmod l$ with another integer l, which is the order of a base point, is also required although the number is a few.

In this paper, the authors give a program to construct elliptic curves such that reduction $\bmod l$ can be computed as mod a pseudo Mersenne prime. The program found elliptic curves $638y^2 = x^3 + 10x^2 + x \bmod p = 2^{256} - 58097$, $82y^2 = x^3 + 18x^2 + x \bmod p = 2^{256} - 507225$, and $3805y^2 = x^3 + 18x^2 + x \bmod p = 2^{256} - 979077$ [1].

Keywords: Elliptic Curve, Elliptic Curve Signature, Modular Multiplication, Pseudo Mersenne Prime

1 INTRODUCTION

Recently, elliptic curve signatures as ECDSA are often used in the TLS communication and block chains. In Europe, it was decided to use an elliptic curve signature in the V2X communication [5]. Elliptic curve signature compared with RSA signature has the advantage that signature generation/ verification is faster and signature length is shorter with same security. However, signature verification in the V2X communication requires further speeding up.

Elliptic curve is a cubic curve given by

$$y^2 = x^3 + ax + b \text{ (Weierstrass form), or}$$
$$By^2 = x^3 + Ax^2 + x \text{ (Montgomery curve),}$$

where x and y are variables. A remarkable feature of elliptic curve is that an operation $+$ is defined[2] in the set of points on E, and the set forms a group for the operation [15].

Dominant processes of the operation $+$, which is explained in Sec.2.2, is modular multiplications

$$X \cdot Y \bmod p \tag{1}$$

for $X, Y \in \mathbb{F}_p = \{0, 1, 2, \cdots, p-1\}$, where p is typically a 256-bit prime. The calculation of (1) is divide into

$$Z \leftarrow \underbrace{X}_{256\text{ bit}} \cdot \underbrace{Y}_{256\text{ bit}}, \tag{2}$$
$$W \leftarrow \underbrace{Z}_{512\text{ bit}} \bmod \underbrace{p}_{256\text{ bit}}. \tag{3}$$

Dominant operation of elliptic curve cryptosystems (ECCs) including elliptic curve signatures is thousands of modular multiplications (1). Therefore, it is important to speed up (2) and (3) to speed up processes of elliptic curve signatures. As explained in Sec.2.4, using Montgomery curve rather than Weierstrass form reduces the number of modular multiplications required for signature generation and verification. Moreover, when a coefficient A of Montgomery curve is 6, 10, 14, and 18, the number of modular multiplications is further reduced.

Karatsuba method, use of high speed multiplier, and parallel implementation speed up integer multiplications. Montgomery reduction [8] that can be applied to arbitrary odd number p is a famous method for reduction $\bmod p$. Also, when p is a pseudo Mersenne prime written as $p = 2^n - k,\ k < 2^{n/2}$, reduction $\bmod p$ can be very efficient.

Curve25519 [1] is a Montgomery curve

$$E_{25519} : y^2 = x^3 + 486662x^2 + x$$

for a pseudo Mersenne prime $p = 2^{255} - 19$. Curve25519 is secure and suitable for high-speed implementation of ECCs and then it has attracted attention in recent years. Some of NIST curves [11], which are elliptic curves for ECCs standardized by NIST, also use pseudo Mersenne prime.

In public key encryptions as ECElGamal and key agreements as ECDH using elliptic curves, required reduction is $\bmod p$ only. On the contrary, in elliptic curve signatures, required reductions are not only $\bmod p$ but also $\bmod l$, where l is the order of a base point.

The order l of a base point on Curve25519 and NIST curves is not pseudo Mersenne prime. Hence, when implementing a signature using these curves, we have to use Montgomery reduction to compute reduction $\bmod l$. However, when implementing an elliptic curve signature by hardware and applying the high-speed reduction for $\bmod p$ and Montgomery

[1]This work was supported by JSPS KAKENHI Grant Number 16K00188.

[2]The operation $+$ is conventionally used for the operation, however, it is different from ordinary addition.

reduction for $\bmod l$ in-creases the hardware scale. Applying Montgomery reduction for both of $\bmod p$ and $\bmod l$ increases computation time. Therefore, it is desirable to be able to compute $\bmod l$ by same way of $\bmod p$.

The purpose of this paper is to make a program to find Montgomery curves such that $\bmod\ l$ can be computed by same way of $\bmod p$ for pseudo Mersenne prime and $A = 6$, 10, 14, and 18, and to give examples of such curves.

Sec.2 explains the definition of elliptic curve, operation $+$, scalar multiplication, coordinate system, secure elliptic curve, and Curve25519. Sec.3 introduces ECDSA that is the most popular elliptic curve signature. Sec.4 introduces efficient reduction methods. Sec.5 is the contribution of this paper. Sec.5 proposes a requirement for elliptic curves to be suitable for ECDSA. Then, Sec.5 makes a program to find elliptic curves that meet the requirement, and gives examples of such curves. Sec.**??** concludes this paper and gives future work.

2 ELLIPTIC CURVE

Sec.2 introduces subjects of elliptic curves required for this paper. For details for subjects of Secs.2.1, 2.2, and 2.3, refer to [15] or [4].

2.1 Definition of Elliptic Curve

Elliptic curve is a cubic curve given by

$$E : y^2 = x^3 + ax + b \text{ (Weierstrass form)} \tag{4}$$

or

$$E : By^2 = x^3 + Ax^2 + x \text{ (Montgomery curve)} \tag{5}$$

with variables x, y. When used in cryptosystems, Montgomery curve (5) is often selected because it can reduce the cost of cryptographic processes.

For a prime p, the set $\mathbb{F}_p$ is defined as

$$\mathbb{F}_p = \{0, 1, 2, \cdots, p-1\}$$

Then, elliptic curve E can be considered on $\mathbb{F}_p$. We consider an elliptic curve

$$E' : y^2 = x^3 + 2$$

on $\mathbb{F}_5$ as an example. For $x, y \in \mathbb{F}_5 = \{0, 1, 2, 3, 4\}$, when

$$y^2 = x^3 + 2 \bmod 5$$

is satisfied, (x, y) is regarded as a point in E' on $\mathbb{F}_5$. For example, $(2, 0)$ is a point in E' on $\mathbb{F}_5$ because

$$0^2 = 2^3 + 2 \bmod 5$$

is satisfied. $(1, 1)$ is not a point in E' on $\mathbb{F}_5$ because

$$1^2 \neq 1^3 + 2 \bmod 5.$$

Executing a program (written for PARI/GP [12]) of Fig. 1, we see that all points in E' on $\mathbb{F}_5$ are

$$\{(2, 0), (3, 2), (3, 3), (4, 1), (4, 4)\}.$$

```
\\Finding points on elliptic curve
{
  p=5;
  for(x=0,p-1,
    for(y=0,p-1,
      if((y^2-(x^3+2))%p==0,
        print([x,y]);
      );
    );
  );
}
```

Figure 1: Program for finding $\mathbb{F}_p$ points on E'

The set adding this set with the point at infinity $\mathcal{O}$ [3] is written as $E'(\mathbb{F}_5)$.

$$E'(\mathbb{F}_5) = \{(2, 0), (3, 2), (3, 3), (4, 1), (4, 4), \mathcal{O}\} \tag{6}$$

The order of $E(\mathbb{F}_p)$, $\#E(\mathbb{F}_p)$, is defined as the number of points in $E(\mathbb{F}_p)$. Hence, the order of $E'(\mathbb{F}_5)$ is 6. The trace of $E(\mathbb{F}_p)$ is defined as an integer t such that

$$\#E(\mathbb{F}_p) = p + 1 - t.$$

When a Montgomery curve $E : By^2 = x^3 + Ax^2 + x$ on $\mathbb{F}_p$ has the trace t, another Montgomery curve $E' : B'y^2 = x^3 + Ax^2 + x$ has the trace t or $-t$. In other words, we have

$$\#E'(\mathbb{F}_p) = \#E(\mathbb{F}_p) \text{ or } 2p + 2 - \#E(\mathbb{F}_p).$$

When $\#E'(\mathbb{F}_p) = 2p + 2 - \#E(\mathbb{F}_p)$, E' is called the twist of E.

Let E be an elliptic curve and L the order of $E(\mathbb{F}_p)$. Then, it is known that

$$p + 1 - 2\sqrt{p} \leq L \leq p + 1 + 2\sqrt{p} \tag{7}$$

is held (Hasse's theorem). That means L is close to p. Conversely, for a prime p and an integer L satisfying (7), Weierstrass form elliptic curve E exists such that $\#E(\mathbb{F}_p) = L$. On the other hand, when E is a Montgomery curve, the order is always a multiple of 4.

2.2 Operation $+$

Let E be an elliptic curve given by Weierstrass form (4) or Montgomery curve (5). Then, the operation $+$ in $E(\mathbb{F}_p)$ is defined as follows.

1. For any $P \in E(\mathbb{F}_p)$,
$$P + \mathcal{O} = \mathcal{O} + P^{4}.$$
2. In the case of $P = (x_1, y_1), Q = (-x_1, y_1) \in E(\mathbb{F}_p)$,
$$P + Q = \mathcal{O}.$$

[3] When considering an elliptic curve in the real plane, $\mathcal{O}$ is intuitively (∞, ∞) and then $\mathcal{O}$ is called "the point at infinity." When also considering an elliptic curve in $\mathbb{F}_p$, $\mathcal{O}$ is called the point at infinity [16, IV.1]. Although $\mathcal{O}$ is a special point, it can be dealt with as normal one in projective coordinate system. Refer to [16, Appendix A] for details.

[4] $\mathcal{O}$ plays a role of zero element in the operation.

Table 1: Computation of $P_i + P_j$ on $E'(\mathbb{F}_5)$

	P_0	P_1	P_2	P_3	P_4	P_5
P_0	P_0	P_1	P_2	P_3	P_4	P_5
P_1	P_1	P_0	P_4	P_5	P_2	P_3
P_2	P_2	P_4	P_3	P_0	P_5	P_1
P_3	P_3	P_5	P_0	P_2	P_1	P_4
P_4	P_4	P_2	P_5	P_1	P_3	P_0
P_5	P_5	P_3	P_1	P_4	P_0	P_2

3. In the case of $P = (x_1, y_1), Q = (x_2, y_2) \in E(\mathbb{F}_p), x_1 \neq x_2$, and $P \neq Q$, $P + Q = (x_3, y_3)$ is computed as

$$\left.\begin{aligned} \lambda &= \frac{y_1 - y_2}{x_1 - x_2}, \\ x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1. \end{aligned}\right\} \quad (8)$$

4. In the case of $P = Q = (x_1, y_1) \in E(\mathbb{F}_p)$, $P + Q = (x_3, y_3)$ is computed as

$$\left.\begin{aligned} \lambda &= \begin{cases} \dfrac{3x_1^2 + a}{2y_1} & E\text{: Weierstrass} \\ \dfrac{3x_1^2 + Ax_1 + 1}{2By_1} & E\text{: Montgomery,} \end{cases} \\ x_3 &= \lambda^2 - 2x_1, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned}\right\} \quad (9)$$

where a, A, and B are coefficients of Eqs. (4) and (5). Eqs.(8) and (9) are called addition formula and doubling formula, respectively.

For any $P, Q, R \in E(\mathbb{F}_p)$, the following are held.

1. $(P + Q) + R = P + (Q + R)$
2. $P + \mathcal{O} = \mathcal{O} + P = P$
3. For $P = (x_1, y_1)$ and $Q = (-x_1, y_1)$, $P + Q = \mathcal{O}$.
4. $P + Q = Q + P$

That the above holds means that $E(\mathbb{F}_p)$ forms a group. This property is very significant, and it is also used for cryptosystems.

Let $E'(\mathbb{F}_5)$ of (6) be written as

$$E'(\mathbb{F}_5) = \left\{ \begin{array}{c} P_0 = \mathcal{O}, P_1 = (2, 0), P_2 = (3, 2), \\ P_3 = (3, 3), P_4 = (4, 1), P_5 = (4, 4) \end{array} \right\}. \quad (10)$$

Then, all results of $P_i + P_j$ are given by Table 1.

2.3 Scalar Multiplication

For a base point $P \in E(\mathbb{F}_p)$ and a natural number n, additions of n terms of P,

$$nP = P + P + \cdots + P$$

is called scalar multiplication. For $P \in E(\mathbb{F}_p)$, the order of P is defined as the smallest positive integer l such that $lP = \mathcal{O}$.

For the order L of $E(\mathbb{F}_p)$ and the order l of $P \in E(\mathbb{F}_p)$, the followings are held (Lagrange's theorem).

1. l is a divisor of L.
2. $LP = \mathcal{O}$.

Let $E'(\mathbb{F}_5)$ be of (10). Then, we see that $2P_1$, $3P_1$, $4P_1$, $\cdots$ are

$$\begin{aligned} 2P_1 &= P_1 + P_1 = \mathcal{O}, \\ 3P_1 &= 2P_1 + P_1 = \mathcal{O} + P_1 = P_1, \\ 4P_1 &= 3P_1 + P_1 = P_1 + P_1 = \mathcal{O}, \\ 5P_1 &= 4P_1 + P_1 = \mathcal{O} + P_1 = P_1, \\ 6P_1 &= 5P_1 + P_1 = P_1 + P_1 = \mathcal{O}, \end{aligned}$$

and $2P_4$, $3P_4$, $4P_4$, $\cdots$ are

$$\begin{aligned} 2P_4 &= P_4 + P_4 = P_3, \\ 3P_4 &= 2P_4 + P_4 = P_3 + P_4 = P_1, \\ 4P_4 &= 3P_4 + P_4 = P_1 + P_4 = P_2, \\ 5P_4 &= 4P_4 + P_4 = P_2 + P_4 = P_5, \\ 6P_4 &= 5P_4 + P_4 = P_5 + P_4 = \mathcal{O}. \end{aligned}$$

Hence, the order of $P_1 \in E'(\mathbb{F}_5)$ is 2, and the order of $P_4 \in E'(\mathbb{F}_5)$ is 6. Also we see Lagrange's theorem holds.

Algorithm 1 (Binary method) and Algorithm 2 (Montgomery reduction) are algorithms for computing a scalar multiplication. Let n be a k-bit integer, and

$$n = (n_{k-1}, n_{k-2}, \cdots, n_0)_2$$

be the binary representation of n. Then, Algorithm 1 takes k doubling formulas and $k/2$ addition formulas on average, and Algorithm 2 takes k doubling formulas and k addition formulas.

Algorithm 1 (Binary method)

Input: $P \in E(\mathbb{F}_p), n = (n_{k-1}n_{k-2}\cdots n_0)_2 \in \mathbb{N}$
Output: $nP \in E(\mathbb{F}_p)$

1. $Q \leftarrow P$
2. **for** $i = k - 2$ **down to** 0
3. $\quad Q \leftarrow 2Q$
4. $\quad$ **if** $n_i = 1$ **then** $Q \leftarrow Q + P$
5. **end for**
6. **return** Q

Algorithm 2 (Montgomery ladder)

Input: $P \in E(\mathbb{F}_p), n = (n_{k-1}n_{k-2}\cdots n_0)_2 \in \mathbb{N}$
Output: $nP \in E(\mathbb{F}_p)$

1. $Q_0 \leftarrow \mathcal{O},\ Q_1 \leftarrow P$
2. **for** $i = k - 1$ **down to** 0
3. $\quad$ **if** $n_i = 0$ **then** $Q_1 \leftarrow Q_0 + Q_1,\ Q_0 \leftarrow 2Q_0$
4. $\quad$ **if** $n_i = 1$ **then** $Q_0 \leftarrow Q_0 + Q_1,\ Q_1 \leftarrow 2Q_1$
5. **end for**
6. **return** Q_0

2.4 Coordinate System

Addition formula (8) and doubling formula (9) needs a division, whose cost is high, to compute λ. Hence, we would like to compute both formulas without division.

For a point (x, y) in the xy coordinate system, another coordinate system that uses X, Y, Z satisfying $x = X/Z$, $y = Y/Z$ to represent (x, y) as (X, Y, Z) is called the projective coordinate system. In the projective coordinate system, addition formula and doubling formula can be computed without division. Hence, the projective coordinate system (or its variants) is generally used in cryptographic implementations. Algorithms 3 and 4 are addition and doubling formulas for Weierstrass form in the projective coordinate system. In these algorithms, Roman face means temporal variables.

Algorithm 3
(Addition for Weierstrass on projective coordinates)

Input: $P = (X_1, Y_1, Z_1), Q = (X_2, Y_2, Z_2) \in E(\mathbb{F}_p)$
Output: $P + Q = (X_3, Y_3, Z_3) \in E(\mathbb{F}_p)$

1. $\mathtt{Y1Z2} \leftarrow Y_1 \cdot Z_2$
2. $\mathtt{X1Z2} \leftarrow X_1 \cdot Z_2$
3. $\mathtt{Z1Z2} \leftarrow Z_1 \cdot Z_2$
4. $\mathtt{u} \leftarrow Y_2 \cdot Z_1 - \mathtt{Y1Z2}$
5. $\mathtt{uu} \leftarrow \mathtt{u}^2$
6. $\mathtt{v} \leftarrow X_2 \cdot Z_1 - \mathtt{X1Z2}$
7. $\mathtt{vv} \leftarrow \mathtt{v}^2$
8. $\mathtt{vvv} \leftarrow \mathtt{v} \cdot \mathtt{vv}$
9. $\mathtt{R} \leftarrow \mathtt{vv} \cdot \mathtt{X1Z2}$
10. $\mathtt{A} \leftarrow \mathtt{uu} \cdot \mathtt{Z1Z2} - \mathtt{vvv} - (\mathtt{R} + \mathtt{R})$
11. $X_3 \leftarrow \mathtt{v} \cdot \mathtt{A}$
12. $Y_3 \leftarrow \mathtt{u} \cdot (\mathtt{R} - \mathtt{A}) - \mathtt{vvv} \cdot \mathtt{Y1Z2}$
13. $Z_3 \leftarrow \mathtt{vvv} \cdot \mathtt{Z1Z2}$

Algorithm 4
(Doubling for Weierstrass on projective coordinates)

Input: $P = (X_1, Y_1, Z_1) \in E(\mathbb{F}_p), a$ of Eq.(4)
Output: $2P = (X_3, Y_3, Z_3) \in E(\mathbb{F}_p)$

1. $\mathtt{XX} \leftarrow X_1^2$
2. $\mathtt{ZZ} \leftarrow Z_1^2$
3. $\mathtt{w} \leftarrow a \cdot \mathtt{ZZ} + 3\mathtt{XX}$
4. $\mathtt{s} \leftarrow 2(Y1 \cdot Z1)$
5. $\mathtt{ss} \leftarrow \mathtt{s}^2$
6. $\mathtt{sss} \leftarrow \mathtt{s} \cdot \mathtt{ss}$
7. $\mathtt{R} \leftarrow Y1 \cdot \mathtt{s}$
8. $\mathtt{RR} \leftarrow \mathtt{R}^2$
9. $\mathtt{B} \leftarrow (\mathtt{X1} + \mathtt{R})^2 - \mathtt{XX} - \mathtt{RR}$
10. $\mathtt{h} \leftarrow \mathtt{w}^2 - 2\mathtt{B}$
11. $X_3 \leftarrow \mathtt{h} \cdot \mathtt{s}$
12. $Y_3 \leftarrow \mathtt{w} \cdot (\mathtt{B} - \mathtt{h}) - 2\mathtt{RR}$
13. $Z_3 \leftarrow \mathtt{sss}$

Let $P = (X, Y, Z)$ be a point on a Montgomery curve in the projective coordinate system. Then, X and Z coordinates of nP can be computed from X and Z coordinates of P using Algorithm 2 (Montgomery ladder) [9]. In this case, Algorithms 5 and 6 are used. For details for these algorithms, refer to [4]. By the way, although omitted, reduction $\bmod p$ is required at every step of Algorithms 3, 4, 5, and 6. For details for them, refer to [2].

Table 2: The cost of Algorithms 3,4,5,6

	Cost	Purpose
Algorithm 3	$14M + 7add$	Addition for Weierstrass
Algorithm 4	$11M + M_a + 12add$	Doubling for Weierstrass
Algorithm 5	$6M + 6add$	Addition for Montgomery
Algorithm 6	$4M + M_{A'} + 4add$	Doubling for Montgomery

Table 3: Cost of scalar multiplication of nP, where n is k-bit

Used curve	Used algorithms	Cost
Weierstrass	Algorithms 1,3,4	$18kM + kM_a + 18k\ add$
Montgomery	Algorithms 2,5,6	$10kM + kM_{A'} + 10k\ add$

We will estimate the cost of theses algorithms in the number of modular multiplications because the cost of modular addition/subtraction is much smaller than one of modular multiplication. Let M, M_a, $M_{A'}$, and add denote a modular multiplication in $\mathbb{F}_p$, a modular multiplication in $\mathbb{F}_p$ with a constant a, a modular multiplication in $\mathbb{F}_p$ with a constant A', and a modular addition/subtraction in $\mathbb{F}_p$, respectively, where a is of (4), and $A' = (A + 2)/4$ for A of (5). Then, the cost of the algorithms are given by Table 2. Note that $M_{A'}$ is for $A' \cdot \mathtt{C}$ at step 7 of Algorithm 6. Also, the cost of scalar multiplication are given by Table 3. We see that the cost of a scalar multiplication on a Montgomery curve is less than one on Weierstrass form.

2.5 Secure Elliptic Curve

The security of ECCs including digital signature depends on the maximum prime factor l of the order $L = \#E(\mathbb{F}_p)$, not the size of p [13]. Attack time against ECCs is roughly proportional to $\sqrt{l}$ and then the larger l is, the more secure ECCs are. Therefore, we have to select elliptic curve E such that

$$L = \#E(\mathbb{F}_p) \text{ has a big prime factor} \tag{11}$$

for ECCs. Also, it is desirable that

$$\begin{gathered} 2p + 2 - L \text{ that is the order of the twist of } E \\ \text{has a big prime factor} \end{gathered} \tag{12}$$

according to [3]. Moreover, we have to select E such that

$$L \neq p, p \pm 1 \tag{13}$$

according to [6], [14].

2.6 Curve25519

Curve25519 is a Montgomery curve

$$E_{25519} : y^2 = x^3 + 486662x^2 + x$$

Algorithm 5
(Addition for Montgomery on XZ coordinates)

Input: $Q_1 - Q_0 = (X_1, Z_1), Q_0 = (X_2, Z_2), Q_1 = (X_3, Z_3)$
Output: $Q_0 + Q_1 = (X_4, Z_4) \in E(\mathbb{F}_p)$

1. $\mathtt{A} \leftarrow X_2 + Z_2$
2. $\mathtt{B} \leftarrow X_2 - Z_2$
3. $\mathtt{C} \leftarrow X_3 + Z_3$
4. $\mathtt{D} \leftarrow X_3 - Z_3$
5. $\mathtt{DA} \leftarrow \mathtt{D} \cdot \mathtt{A}$
6. $\mathtt{CB} \leftarrow \mathtt{C} \cdot \mathtt{B}$
7. $X_4 \leftarrow Z_1 \cdot (\mathtt{DA} + \mathtt{CB})^2$
8. $Z_4 \leftarrow X_1 \cdot (\mathtt{DA} - \mathtt{CB})^2$

Algorithm 6
(Doubling for Montgomery on XZ coordinates)

Input : $R = (X_1, Z_1), A' = (A+2)/4$ for A in Eq.(7)
Output: $2R = (X_4, Z_4) \in E(\mathbb{F}_p)$

1. $\mathtt{A} \leftarrow X_1 + Z_1$
2. $\mathtt{AA} \leftarrow \mathtt{A}^2$
3. $\mathtt{B} \leftarrow X_1 - Z_1$
4. $\mathtt{BB} \leftarrow \mathtt{B}^2$
5. $\mathtt{C} \leftarrow \mathtt{AA} - \mathtt{BB}$
6. $X_4 \leftarrow \mathtt{AA} \cdot \mathtt{BB}$
7. $Z_4 \leftarrow C \cdot (\mathtt{BB} + A' \cdot \mathtt{C})$

with $p = 2^{255} - 19$ [1]. The order $L = \#E_{25519}(\mathbb{F}_p)$ is

$$L = 2^2 \cdot l,$$
$$l = 2^{252} + 27742317777372353535851937790883648493,$$

where l is a 253-bit prime. Curve25519 meets the security requirement in Sec.2.5.

Curve 25519 has been applied in many cryptographic libraries such as NaCl [10], and Curve 25519 was added to Special Publication 800-186, which specifies the approved elliptic curve used by the US federal government by NIST in 2017.

3 ECDSA

ECDSA is a digital signature using an elliptic curve. ECDSA consists of *system parameter* held by all users, *key generation* for generating each user's (private key, public key), *signature generation* for generating a signature using a user A's secret key, and *signature verification* for verifying the signature using A's public key.

System parameter
A sufficiently large (e.g. 256-bit) prime p, an elliptic curve E such that $E(\mathbb{F}_p)$ meets the security requirements (11), (12), and (13), and a base point $G \in E(\mathbb{F}_p)$ of which the order is l are selected. Also a hash function $H : \{0,1\}^* \rightarrow \{0, 1, 2, ..., l-1\}$ is selected. (p, l, E, G, H) is the system parameter.

Key generation
User A choose $s \in [1, l-1]$ at random, and computes $Y = sG$ (scalar multiplication) in $E(\mathbb{F}_p)$. Then, s and Y are A's private key and public key, respectively.

Signature generation
User A generates a signature of a message $m \in \{0,1\}^*$ as follows.

1. Computing $m' = H(m)$.
2. Choosing $r \in [1, l-1]$ at random, and compute
$$U = \underbrace{rG}_{\text{scalar mul. on } E(\mathbb{F}_p)} = (u_x, u_y),$$
$$u = u_x \bmod l.$$
3. Using the secret key s to compute
$$v = r^{-1}(m' + su) \bmod l.$$
4. The pair (u, v) is the signature of m.

Signature verification
A recipient of the message m with signature (u, v) verifies the signature as follows.

1. Computing $m' = H(m)$.
2. Computing $d = v^{-1} \bmod l$.
3. Computing $U' = \underbrace{(m'd)G}_{\text{scalar mul. on } E(\mathbb{F}_p)} + \underbrace{(ud)Y}_{\text{scalar mul. on } E(\mathbb{F}_p)}$.
4. Computing $u' = (\text{the } x \text{ coordinate of } U') \bmod l$.
5. If $u = u'$ then the signature is accepted, and if $u \neq u'$ then it is rejected.

Thus, the dominant processes of ECDSA is scalar multiplications in $E(\mathbb{F}_p)$ [5]. Signature generation of ECDSA takes one scalar multiplication and signature verification of ECDSA takes two scalar multiplications. Therefore, we see that

in order to speed up processes of ECDSA,
it is important to speed up scalar multiplication.

As seen Table 3, using not Weierstrass form but Montgomery curve reduces the number of modular multiplications required for a scalar multiplication.

4 MODULAR REDUCTION

In order to speed up ECCs including ECDSA, it is important not only to reduce the number of modular multiplications but also to reduce the cost of one modular multiplication. This section introduces efficient reduction methods.

4.1 Montgomery Reduction

The Montgomery reduction (Algorithm 7) [8] is a method for efficiently calculating $X \bmod N$ for general odd number N and X given in Montgomery representation[6].

[5]The dominant process of not only ECDSA but also all ECCs is scalar multiplications.

[6]For Montgomery representation, refer to [8] or [4].

Algorithm 7 (Montgomery Reduction)

Input: odd number N of n bits, $R = 2^n$, $N' = (-N^{-1}) \bmod R$, natural number $u < RN$
Output: $u \bmod N$ in Montgomery representation

1. $\mathtt{t} \leftarrow u$
2. $\mathtt{k} \leftarrow \mathtt{t}N' \bmod R$
3. $\mathtt{t} \leftarrow \mathtt{t} + \mathtt{k}N$
4. $\mathtt{t} \leftarrow t/R$
5. **if** $\mathtt{t} \geq N$ **then** $\mathtt{t} \leftarrow \mathtt{t} - N$
6. **return** $\mathtt{t}$

4.2 Reduction modulo Pseudo Mersenne Prime

When a prime p is written as

$$p = 2^n - k,\ k < 2^{n/2},$$

it is called pseudo Mersenne prime. For pseudo Mersenne prime p, reduction $\bmod p$ can be computed at high speed using by Algorithm 8 [7]. Notice that $u/2^n$ in step 1 and $v/2^n$ in step 3 are integer divisions and then they are performed by shift operations.

Algorithm 8 (Reduction mod pseudo Mersenne prime)

Input: prime $p = 2^n - k$ $(k < 2^{n/2})$, integer $0 \leq u \leq (p-1)^2$
Output: $u \bmod p$

1. $\mathtt{u0} \leftarrow u \bmod 2^n$, $\mathtt{u1} \leftarrow u/2^n$
2. $\mathtt{v} \leftarrow \mathtt{u1} \cdot k + \mathtt{u0}$
3. $\mathtt{v0} \leftarrow \mathtt{v} \bmod 2^n$, $\mathtt{v1} \leftarrow \mathtt{v}/2^n$
4. $w \leftarrow \mathtt{v1} \cdot k + \mathtt{v0}$
5. **if** $w \geq p$ **then** $w \leftarrow w - p$
6. **return** w

5 CONTRIBUTIONS

5.1 Program to Search Elliptic Curve Suitable for ECDSA

The purpose of this paper is to make a program to search for elliptic curves that is secure and suitable for high-speed implementation of ECDSA (especially by hardware implementation), and to give examples of such an elliptic curves. Specifically, we will search curves that meet the following requirements.

Elliptic Curve Requirements to Search

1. According to Table 3, scalar multiplication in Montgomery curve takes fewer modular multiplications than scalar multiplication in Weierstrass form. Hence, Montgomery curve is selected.

2. According to Table 3 again, scalar multiplication in Montgomery curve with $A' = 1, 2, 3, 4, 5$, that is, the coefficient $A = 2, 6, 10, 14, 18$, requires fewer modular multiplications than other As [7]. Therefore, we take $A = 2, 6, 10, 14, 18$.

3. Prime p is typical 256-bit. Moreover, p is a pseudo Mersenne prime $p = 2^n - k$ because of efficient reduction $\bmod p$. For convenience of execution time, set the range of k to $k \leq 2^{20}$.

4. To meet the security requirement (11), the order $L = \#E(\mathbb{F}_p)$ is as $L = 4l, 8l, 16l$, where l is a prime. Notice there is a point $P \in E(\mathbb{F}_p)$ whose order is l.

5. To meet the security requirement (12), $L' = 2p + 2 - L$ is as $L' = 4l', 8l', 16l'$, where l' is a prime.

6. To meet the security requirement (13), curves such as $L = p, p \pm 1$ is removed.

7. L is written as $L = 2^n - k',\ k' < 2^{n/2}$. Then, a reduction $\bmod l$ is computed by the algorithm proposed in Sec.5.2, which is as efficient as reduction mod a pseudo Mersenne prime.

Note Curve25519 also meets the requirements 1, 4, 5, and 7, and Curve25519 adopts not 256-bit prime but 254-bit for a prime field. Curce25519 does not consider the requirements 2 and 7. Also refer to Sec.5.3.

The authors made a program as Fig. 2 in PARI/GP to search elliptic curves meeting the requirements. The program is straightforward and then it may be easy for some readers to make a similar program. But, giving the program makes all readers (especially PARI/GP users) generate good curves. Notice that the program output only a prime p, a coefficient A of Montgomery curve, the order L of $E(\mathbb{F}_p)$, and the order l of a base point. At the moment, it is necessary to manually find another coefficient B of Montgomery curve, generate a base point whose order is l, and check $L \neq p, p-1$.

This program is briefly explained. The line

```
e=ellinit([0,A,0,1,0]);
```

sets (Montgomery) elliptic curve $E : y^2 = x^3 + Ax^2 + x$ to `e`. The function `ellap(e,p)` outputs the trace of $E(\mathbb{F}_p)$. Thus, `Num1` is the order of $E(\mathbb{F}_p)$, and `Num2` is the order of the twist of $E(\mathbb{F}_p)$. `isprime` is a prime decision function. `write` is an output function to text.

By this program, the following elliptic curves are found.

1. $p = 2^{256} - 58097$,
   $E_{S1} : 638y^2 = x^3 + 10x^2 + x$,
   base point $P = (11, 2)$,
   $L = 2^{256} - k'$, where k' is 125-bit integer
   $k' = 2518136338042871045307996739901786932 8$,
   $l = L/16$, which is 252-bit prime.

[7] Selecting an elliptic curve with appropriate coefficients to reduce high cost multiplication into low cost addition is one of fast implementation techniques of ECCs [4, Sec. 13.2.1c]. For Montgomery curve, if $A' = 1$ then the multiplication $A' \cdot \mathtt{C}$ at step 7 of Algorithm 4 is free. If $A' = 2$ then the multiplication is performed by an addition $\mathtt{C} + \mathtt{C}$. As well, if $A' = 3$ or 4 then the multiplication is performed by two additions.

```
\\Checking pseudo Mersenne prime
check_mer(p)={
  local(n,c,k);
  n=0;
  c=0;
  while(c==0,
    n++;
    if(2^(n-1)<=p && p<2^n,c=1);
  );
  if(2^n-p < sqrt(2^n),
    k=floor(log(2^n-p)/log(2)+1);
    return([n,k]),
    return(0));
}

\\Main program
{
  count=0;
  for(a=0,4,
    A=4*a+2;
    e=ellinit([0,A,0,1,0]);
    for(k=1,2^20,
      print([a,k,count]);
      p=2^256-k;
      if(isprime(p)==1,
        t=ellap(e,p);
        num1=p+1-t;
        num2=p+1+t;
        Num1=num1;
        Num2=num2;
        check1=0;
        check2=0;
        if(num1%2==0,num1=num1/2;check1=1);
        if(num1%2==0,num1=num1/2;check1=2);
        if(num1%2==0,num1=num1/2;check1=3);
        if(num1%2==0,num1=num1/2;check1=4);
        if(num2%2==0,num2=num2/2;check2=1);
        if(num2%2==0,num2=num2/2;check2=2);
        if(num2%2==0,num2=num2/2;check2=3);
        if(num2%2==0,num2=num2/2;check2=4);
        isprime_num1=isprime(num1);
        isprime_num2=isprime(num2);
        if(t!=0 && isprime_num1==1
                && isprime_num2==1
                && (check_mer(Num1)!=0
                || check_mer(Num2)!=0),
          if(check_mer(Num1)!=0,
            count++;
            write("ijis.txt",k","A","Num1",
            "num1);
          );
          if(check_mer(Num2)!=0,
            count++;
            print("A="A);
            write("ijis.txt",k","A","Num2",
            "num2);
          );
        );
      );
    );
  );
}
```

Figure 2: Proposed program to find elliptic curves suitable for ECDSA

2. $p = 2^{256} - 507225$,
   $E_{S2} : 82y^2 = x^3 + 18x^2 + x$,
   base point $P = (2, 1)$,
   $L = 2^{256} - k'$, where k' is 127-bit integer
   $k' = 134184981501621384111934924743103436264$,
   $l = L/8$, which is 253-bit prime.

3. $p = 2^{256} - 979077$,
   $E_{S3} : 3805y^2 = x^3 + 18x^2 + x$,
   base point $P = (20, 2)$,
   $L = 2^{256} - k'$, where k' is 126-bit integer
   $k' = 67240641251824776802983670794157366424$,
   $l = L/8$, which is 253-bit prime.

For the convenience of time, the authors set the search range to $k < 2^{20}$, however, if the search range is expanded, more appropriate elliptic curve may be found.

5.2 Proposed Modular Reduction

This section proposes an algorithm (Algorithm 9) [8] similar to Algorithm 8 for computing a reduction $\bmod l$ for a prime l such that $L = 2^m l$ is written as $L = 2^n - k,\ k < 2^{n/2}$.

Notice $\mathtt{v}$, $\mathtt{v0}$ and k are multiples of 2^m. Thus, $\mathtt{w}$ is also a multiple of 2^m. As well,

$$\mathtt{x} \text{ is a multiple of } 2^m. \tag{14}$$

From step 2 to 7 is same as Algorithm 8 and then we see

$$\mathtt{x} = 2^m u \bmod 2^m l. \tag{15}$$

By (14) and (15), we see $\mathtt{x}/2^m = u \bmod l$.

Note that $\mathtt{v}/2^n$ in step 2, $\mathtt{w}/2^n$ in step 4, and $\mathtt{x}/2^n$ in step 7 are integer divisions and then they are performed by shift operations, and $2^m u$ in step 1 is also performed by a shift operation.

Proposed Algorithm 9

Input : integer l such that $2^m l = 2^n - k$ $(k < 2^{n/2})$, integer $0 \le u \le (l-1)^2$
Output : $u \bmod l$

1. $\mathtt{v} \leftarrow 2^m u$
2. $\mathtt{v0} \leftarrow \mathtt{v} \bmod 2^n$, $\mathtt{v1} \leftarrow \mathtt{v}/2^n$
3. $\mathtt{w} \leftarrow \mathtt{v1} \cdot k + \mathtt{v0}$
4. $\mathtt{w0} \leftarrow \mathtt{w} \bmod 2^n$, $\mathtt{w1} \leftarrow \mathtt{w}/2^n$
5. $\mathtt{x} \leftarrow \mathtt{w1} \cdot k + \mathtt{w0}$
6. **if** $\mathtt{x} \ge 2^m l$ **then** $\mathtt{x} \leftarrow \mathtt{x} - 2^m l$
7. $y \leftarrow \mathtt{x}/2^m$
8. **return** y

5.3 Searched Elliptic Curves v.s. Curve25519

Elliptic curves searched in this paper and Curve 25519 meet the security requirement in Sec.2.5. Both of them adopt pseudo Mersenne prime and then reductions $\bmod p$ for them are efficiently computed. However, computation of reduction $\bmod p$ may be more efficient for Curve 25519 on CPU with small words because k of $p = 2^n - k$ is smaller,

[8] Although the authors do not know whether Algorithm 9 is already known, it may be already known because Algorithm 9 is almost same as Algorithm 8.

The coefficient of searched elliptic curves are $A = 10, 18$ ($A' = 3, 5$), on the other hand, one of Curve 25519 is $A = 486662$ ($A' = 121666$). Notice that when $A' = 2$, a product with A' is reduced to an addition. Hence, the cost of a scalar multiplication with Curve25519 is $10kM + kM_{A'} + 10k\ add$, and one with the searched curve is $10kM + 11k\ add$ by Table 3. In general add is smaller than $M_{A'}$.

A reduction $\bmod l$ can be computed with Algorithm 9 for searched curves, on the other hand, it cannot be for Curve25519. Therefore, ECDSA adopting searched curves is expected to be faster and implemented more efficiently (when it is implemented hardware or CPU with small words) compared with the ECDSA adopting Curve25519.

6 CONCLUSION

This paper searched three elliptic curves suitable for ECDSA. In these curves, not only the reduction $\bmod p$ but also the reduction $\bmod l$ can be computed at high speed, where p is of $\mathbb{F}_p$ and l is the order of a base point. and doubling is faster because of a coefficient of curves $A = 10$ or 18. ECDSA adopting the searched curves has the same security as ECDSA adopting Curve 25519, and it can process faster.

The authors would like to evaluate implementation results as future work. Also, they would like to extend the search range of the suggested program in Fig. 2 and to execute it.

REFERENCES

[1] D. J. Bernstein, "Curve25519: New Diffie-Hellman speed records," PKC 2006, LNCS 3958, pp. 207–228, Springer (2006).

[2] D. J. Bernstein and T. Lange, Explicit-formulas database, https://hyperelliptic.org/EFD/.

[3] D. J. Bernstein and T. Lange, Safecurves: Choosing safe curves for elliptic-curve cryptography, https://safecurves.cr.yp.to.

[4] H. Cohen and G. Frey, editors, Handbook of Elliptic and Hyperelliptic Curve Cryptography, Chapman and Hall/CRC (2005).

[5] ETSI, Etsi ts 103 097 v1.1.1 intelligent transport systems (its); security; security header and certificate formats (2013).

[6] G. Frey and H.-G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," Matheematics of Computation, Vol. 62, No. 206, pp. 865–874 (1994).

[7] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography, CRC Press (1997).

[8] P. L. Montgomery, "Modular multiplication without trial division," Matheematics of Computation, Vol. 44, No. 170, pp. 519–521 (1985).

[9] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," Matheematics of Computation, Vol. 48, No. 177, pp. 243–264 (1987).

[10] Nacl: Networking and cryptography library, https://pari.math.u-bordeaux.fr.

[11] NIST, Nist: Fips 186-2 digital signature standard, https://csrc.nist.gov/csrc/media/publications/fips /186/3/archive/ 2009-06-25/documents/fips_186-3.pdf.

[12] PARI/GP, https://pari.math.u-bordeaux. fr.

[13] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance,"IEEE Transactions on Information Theory, Vol. 24, No. 1, pp. 106–110 (1978).

[14] H.-G. Rück, "On the discrete logarithm in the divisor class group of curves," Matheematics of Computation, Vol. 68, No. 226, pp. 805–806 (1999).

[15] J. H. Silverman, The arithmetic of elliptic curves, Springer-Verlag New York (1985).

[16] J. Tate and J. H. Silverman, Rational points on elliptic curves. Springer-Verlag (1992).

(Received October 21, 2018)
(Revised April 8, 2019)

Masaaki Shirase Masaaki Shirase received the M.S. and Ph.D. degrees in Information Science from Japan Advanced Institute of Science and Technology (JAIST) in 2003 and 2006, respectively. He is currently an Associate Professor in the School of Systems Information Science at Future University Hakodate. His research interests are algorithm and implementation of cryptography.