# An Improved Ternary Montgomery Ladder Algorithm on Elliptic Curves over GF($3^m$)

Shuang-Gen Liu, Rong-Rong Wang, Yun-Qi Li, and Can-Liang Zhai
*(Corresponding author: Shuang-Gen Liu)*

School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications
Xi'an 710121, China
(Email: liusgxupt@163.com)

*(Received Jan. 15, 2018; Revised and Accepted Apr. 21, 2018; First Online Jan. 11, 2019)*

## Abstract

In this paper, we propose a new scalar multiplication algorithm on elliptic curves over GF($3^m$). It combines original Montgomery ladder algorithm and the ternary representation of the scalar, which makes full use of cubing. In addition, in order to improve performance, we have presented new composite operation formulas which are $2P_1+P_2$ and $3P_1$, and applied them to the improved scalar multiplication algorithm. Based on the original Montgomery ladder algorithm, it can resist Simple Power Attack (SPA). In the experimental analysis, we set the ratio of inversion and multiplication to a dynamic value. Results show that with respect to previous algorithm, the average efficiency of proposed scalar multiplication algorithm is increased by 7.8 % to 11.3 % in affine coordinate, and 4.0 % to 17.9 % in projective coordinate.

*Keywords: Characteristic Three; Composite Formulas; Elliptic Curve Cryptography; Montgomery Ladder Algorithm; Scalar Multiplication*

## 1 Introduction

Elliptic Curve Cryptography (ECC) was introduced independently by Koblitz [9] and Victor Miller [16] in around 1985. Its key size is smaller than RSA with the equivalent security, for example, elliptic curve with the key of 160 bits is competitive with RSA with the key of 1024 bits. This can be especially an advantage for applications where resources are limited, such as smart cards, embedded devices and mobile phones. Its safety is based on the difficulty of elliptic curve discrete logarithm problem (ECDLP). ECC is used for encryption, decryption, digital signature and verification [5, 7, 8, 18, 22, 28]. The factors that affect the execution rate of ECC algorithm are generally as follows: The choice of coordinates, scalar multiplication, the selection of elliptic curves. One of the decisive factors is the calculated rate of scalar multiplication. Scalar multiplication, defined as $[k]$P=P+P+...+P, where $k$ is an integer and P is an elliptic curve point, is a major and time-consuming operation in ECC. Scalar multiplication operations can be divided into two layers: the top layer and the bottom layer. Among them, the top layer operation is basic point operation on elliptic curve, such as point addition and point doubling, and the bottom layer operation is underlying field operation, such as inverse, multiplication, squaring, and so on.

In recent years, GF($2^m$)-ECC and GF(p)-ECC have been well studied. There are a lot of methods to improve the elliptic curve scalar multiplication, such as double-and-add [11], non-adjacent form (NAF) [20] and so on, while GF($3^m$)-ECC have been less studied due to their efficiency factors. So the research of fast and security scalar multiplication on the elliptic curve over GF($3^m$) has become one of the hot research topics. GF($3^m$)-ECC, as a special type of GF($p^m$)-ECC, has some properties of fast computation similar to GF($2^m$)-ECC, but it has own special properties and is suitable as a carrier of secure password algorithm [12, 19, 24, 26].

In 1987, Montgomery [17] proposed a fast algorithm for calculating the elliptic curve scalar multiplication $k$P that resists Simple Power Attack(SPA) attacks. SPA is a type of side channel attack proposed by literature [10]. The basic idea is: Integer $k$ is expanded into binary form, which is computed cyclically from left to right. And there are one point doubling and one point addition operations in each cycle. Because of the same computational pattern and cost in every loop iteration, this algorithm prevents SPA. However, the original Montgomery ladder algorithm has the demerit of slow performance. In this paper, a scalar $k$ is represented as ternary form instead of binary form. The length of the ternary expression is shorter than binary expression. Based on Montgomery's idea, a new algorithm was proposed by Lopez and Dahab [15] in 1999, and it was used for calculating the elliptic curve scalar multiplication over GF($2^m$). By using a new set of point addition and point doubling calculation formulas, every iteration requires only the x-coordinate of the point to be calculated, and the y-coordinate is restored

at the end of the algorithm. Smart and Westwood [27] first pointed out that ordinary elliptic curves over finite fields of characteristic three is an alternative for implementing elliptic curve cryptosystems, and it is at most 50% slower than the equivalent system over finite fields of characteristic two. After that, these elliptic curves were extensively studied by many papers [3, 4, 12, 27]. The literature [29] proposed point addition and point doubling calculation formulas that omit the calculation of the Y-coordinate, and remove the inverse operation, thereby improving the Montgomery algorithm over GF($3^m$). The literature [3] improved further point addition, doubling and tripling operation over GF($3^m$). In 2013, Gu *et al.* [4] gave the reason why the Montgomery ladder algorithm performs worse over ternary fields than binary fields. In 2015, Zhou *et al.* [25] deduced a formula of calculating $3^k$ P directly under the affine coordinates. Yu *et al.* [30] optimized Projective Montgomery Algorithm over the finite field with characteristic of 3 by using co-Z tricks, and the Y coordinate is not calculated in the middle of the loop. Robert and Negre [1] first presented thirding point formula together with our third-and-add and parallel approaches for scalar multiplication.

Our contributions in this paper are divided three levels:

- We review researches about scalar multiplication over GF($2^m$), GF(p) and GF($3^m$) in recent years, and related theory about ECC.
- Different from original Montgomery ladder algorithm, an improved ternary Montgomery ladder algorithm over GF($3^m$) is proposed. Furthermore, in order to increase the speed of scalar multiplication, we develop composite operation formulas in the underlying field.
- The proposed algorithm can be applied in elliptic curve cryptography. This algorithm has many advantages over the existing ones, and it has better performance and higher security naturally.

The remainder of this paper is organized as follows. The next section reviews the necessary background for arithmetic on elliptic curves over GF($3^m$), ordinary ternary form of $k$, and related scalar multiplication. In Section 3, we present the improved ternary Montgomery ladder algorithm. Additionally, we derive composite operation formulas which are $2P_1 + P_2$ and $3P_1$. In Section 4, we give some comparison with other algorithms under different coordinate system. Finally, in Section 5, we draw some concluding remarks.

# 2 Background

## 2.1 Elliptic Curve Cryptography

An elliptic curve E over a finite field $K$ is defined by the equation

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \quad (1)$$

where $a_1, a_2, a_3, a_4, a_6 \in K$, and$\Delta \neq 0$, the $\Delta$ is discriminant of E.

When the characteristic of $K$ is equal to 3, we use the non-supersingular form of an elliptic curve given for $a \neq 0$ by

$$y^2 = x^3 + ax^2 + b. \quad (2)$$

where $a, b \in K$, and $\Delta = -a^3b \neq 0$.

$P_1 = (x_1, y_1) \neq O$ and $P_2 = (x_2, y_2) \neq O$ is two different points on the elliptic curve, then the sum of them is $P_3 = (x_3, y_3)$ computed by

$$x_3 = \lambda^2 - x_1 - x_2 - a, y_3 = \lambda(x_1 - x_3) - y_1, \quad (3)$$

where $\lambda = \dfrac{y_2 - y_1}{x_2 - x_1}$.

The doubling of the point $P_1 = (x_1, y_1) \neq$ O is the point $P_3 = (x_3, y_3)$ given by

$$x_3 = \lambda^2 + x_1 - a, y_3 = \lambda(x_1 - x_3) - y_1, \quad (4)$$

where $\lambda = \dfrac{ax_1}{y_1}$.

Both doubling and addition formulae require $1I+2M+1S$, where $I, M, S$ is the cost of a field inverse, multiplication and squaring, respectively.

## 2.2 Elliptic Curve over Fields of Characteristic Three

For elliptic curves over fields of characteristic three, we know some basic facts [27]: polynomial g(z), as an element in the field $F_{3^m} = F_3[z]/M(z)$, the cubing rule can be shown as follows:

$$g(z)^3 = g(z^3) \pmod{\text{M(z)}}.$$

The square uses the same algorithm as multiplication, so it is regarded as equivalent to multiplication. Multiplying and squaring field elements are similar in complexity and are not too expensive, but more costly than cubing. The computing speed of cubing is at least ten times faster than that of multiplication or squaring. Since the characteristic of the elliptic curve is 3, so we can simplify cubic computation with Frobenius self-homomorphism [6], and computational overhead of cubic can be negligible, compared to other operations. Addition and subtraction can be ignored as well [2]. Although, the cubing in ternary fields can be implemented by previous efficient methods, the original Montgomery ladder algorithm do not make full use of cubing [4]. In Section 3, we propose an improved ternary Montgomery ladder algorithm. Here, the ternary expression is useful to reduce the length of the scalar representation.

A curve of the form $y^2 = x^3 + ax^2 + b$ is used in this paper, and it has a point of order three and the order of the group is divisible by three. It needs to meet some requirements, for example, there is group order $3 \times p$, where p is a prime, and it can be performed in the subgroup of size p.

### 2.3 Ordinary Ternary Form

According to the traditional division algorithm, An arbitrary positive integer $k$ is expressed as $k=(k_{n-1},\cdots,k_1,k_0)_3$, where $k_{n-1}$=1 or 2, and $k_i \in \{0,1,2\}, i=0,1,\cdots,n-2$.

**Algorithm 1** Ordinary Ternary Form

**Input:** A positive integer $k$
**Output:** $k=(k_{n-1},\cdots,k_1,k_0)_3$, with $k_{n-1}$=1 or 2 and $k_i \in \{0,1,2\}$.
$i \leftarrow 0$
**while** $k>0$ **do**
  **if** $k \mod 3 = 2$ **then**
    $k_i \leftarrow 2$
    $k = \lfloor k/3 \rfloor$
  **end if**
  **if** $k \mod 3 = 1$ **then**
    $k_i \leftarrow 1$
    $k = \lfloor k/3 \rfloor$
  **end if**
  **if** $k \mod 3 = 0$ **then**
    $k_i \leftarrow 0$
    $k = k/3$
  **end if**
  $i \leftarrow i+1$
**end while**
**Output k.**
End

Algorithm 1 correctness:

1) The result of $k$ mod 3 in this algorithm is only 2, 1 and 0, and the ratio of 2, 1, and 0 is 1/3 [13], on average.

2) A branch is always performed in the loop, so $k$ will certainly continue to decrease after $k \leftarrow k/3$, the program ends the loop when the final result of $k$ is 0. Therefore,any positive integer $k$ will certainly be turned into an ordinary ternary string, after the circular execution of Algorithm 1.

**Example 1.** A positive integer $k = 520$

$i \leftarrow 0$
$k_0 \leftarrow 1, k = 173, i \leftarrow 1$
$k_1 \leftarrow 2, k = 57, i \leftarrow 2$
$k_2 \leftarrow 0, k = 19, i \leftarrow 3$
$k_3 \leftarrow 1, k = 6, i \leftarrow 4$
$k_4 \leftarrow 0, k = 2, i \leftarrow 5$
$k_5 \leftarrow 2, k = 0, i \leftarrow 6$
**Output** $k = \{2,0,1,0,2,1\}$

### 2.4 Scalar Multiplication

#### 2.4.1 Ordinary Ternary Form Scalar Multiplication

Ordinary ternary form scalar multiplication is expressed as left-to-right form, and Algorithm 2 describes corresponding elliptic curve scalar multiplication.

$$kP = \sum_{i=0}^{n-1} k_i 3^i P = 3(\cdots 3(3k_{n-1}P + k_{n-2}P) + \cdots) + k_0 P.$$

**Algorithm 2** Ordinary ternary form scalar multiplication algorithm

**Input:** $P = (x,y) \in E(GF(2^m))$, and $k = (k_{n-1}, k_{n-2}, \cdots, k_1, k_0)_3$
**Output:** $Q = kP \in E(GF(2^m))$.
$R_0 \leftarrow O$
**for** $i = n-1, \cdots, 0$ **do**
  $R_0 = 3R_0$
  $R_1 = R_0 + P$
  $R_2 = R_0 + 2P$
  $R_0 = R_{k_i}$
**end for**
Return $Q = R_0$
End

The algorithm requires ($n$)-time triple, ($n$)-time double and 2($n$)-time addition. Each loop performs the same point operation whatever the key bit is, so attacker can not guess the value of scalar $k$ from power trace of point multiplication, *i.e.*, Algorithm 2 can resist SPA.

#### 2.4.2 A Left-to-Right Montgomery Ladder

The well-known Montgomery ladder for speeding up the scalar multiplication, which was initially proposed and utilized for Montgomery form elliptic curves [17], can be adapted to Weierstrass form curves. Algorithm 3 describes the classical left-to-right Montgomery ladder approach for point multiplication [21].

The algorithm requires ($n$-1)-time double and ($n$-1)-time addition. Each loop performs one point doubling and one point addition operations, so this algorithm prevents SPA. Furthermore, given a base point P, and there is no change in the difference between the input points $R_0$ and $R_1$, *i.e.*, $R_1 - R_0 = P$. This algorithm can also be applied to elliptic curve over GF($3^m$) [30].

## 3 Proposed Algorithm

In this part, we propose a new efficient scalar multiplication algorithm based on the Montgomery ladder algorithm. The scalar $k$ is represented as a ternary form, and the new algorithm is extended to elliptic curves over GF($3^m$).

**Algorithm 3** Left-To-Right Montgomery Ladder Algorithm

**Input:** $P = (x, y) \in E(GF(2^m))$, and $k = (1, k_{n-2}, \cdots, k_1, k_0)_2$
**Output:** $Q = kP \in E(GF(2^m))$.
$R_0 = P$; $R_1 = 2P$
**for** $i = n-2, \cdots, 0$ **do**
  **if** $k_i = 1$ **then**
    $R_0 = R_0 + R_1$; $R_1 = 2R_1$
  **end if**
  **if** $k_i = 0$ **then**
    $R_1 = R_0 + R_1$; $R_0 = 2R_0$
  **end if**
**end for**
Return $Q = R_0$
End

## 3.1 The Improved Ternary Montgomery Ladder Algorithm

Given a positive integer $k$, and it is expressed as ternary form $k = 3^{n-1}k_{n-1} + \cdots + 3k_1 + k_0$, where $k_{n-1}$=1 or 2.

**Definition 1.** *The value of scalar multiplication is stored in $R_0$. We define $R_0^{(i)} = (\sum_{j=1}^{i} \mathrm{k}_{n-j} 3^{i-j})P$ as the value of $R_0$ at the end of the (i-1)-round loop in the algorithm, where $1 \le i \le n$, $j \le i$ and $R_0^{(i)}$ is a point on the elliptic curve. Especially, for $i = 1$, $R_0^{(1)} = (\sum_{j=1}^{1} \mathrm{k}_{n-1} 3^0)P = k_{n-1}P$ is an initial value in the algorithm.*

Similarly, $R_1^{(i)}$ is defined as $R_1^{(i)} = (\sum_{j=1}^{i} k_{n-j} 3^{i-j})P$, and it also holds $R_1^{(i)} = R_0^{(i)} + P$. Then, $R_0^{(i+1)}$and $R_1^{(i+1)}$ are computed by using $R_0^{(i)}$ and $R_1^{(i)}$, and they depend on the value of $k_{n-i-1}$, as follows:

$$\begin{cases} \text{if } k_{n-i-1} = 0, & R_0^{(i+1)} = 3R_0^{(i)}, \\ & R_1^{(i+1)} = 2R_0^{(i)} + R_1^{(i)} \\ \text{if } k_{n-i-1} = 1, & R_0^{(i+1)} = 2R_0^{(i)} + R_1^{(i)}, \\ & R_1^{(i+1)} = 2R_1^{(i)} + R_0^{(i)} \\ \text{if } k_{n-i-1} = 2, & R_0^{(i+1)} = 2R_1^{(i)} + R_0^{(i)}, \\ & R_1^{(i+1)} = 3R_1^{(i)} \end{cases} \quad (5)$$

Therefore, in the calculation of $R_0^{(i+1)}$ and $R_1^{(i+1)}$, two composite operations $2P_1 + P_2$ and $3P_1$ are involved, where $P_1$ and $P_2$ are $R_0^{(i)}$ or $R_1^{(i)}$. $R_1^{(i)} - R_0^{(i)} = P$ is still valid. Algorithm 4 describes the improved ternary Montgomery ladder algorithm over finite fields of characteristic three, and in the following, the algorithm is verified by giving an example.

Notice that at each iteration of Algorithm 4, the variable $R_0$ is updated as

$$R_0 = \begin{cases} 3R_0, & \text{if } k_i = 0 \\ 2R_0 + R_1, & \text{if } k_i = 1 \\ 2R_1 + R_0, & \text{if } k_i = 2 \end{cases} \quad (6)$$

**Algorithm 4** The Improved Ternary Montgomery Ladder Algorithm over GF($3^m$)

**Input:** $P = (x, y) \in E(GF(3^m))$, and $k = (k_{n-1}, k_{n-2}, \cdots, k_1, k_0)_3$, where $k_{n-1} = 1$ or 2
**Output:** $Q = kP \in E(GF(3^m))$.
$R_0 = k_{n-1}P$, $R_1 = (k_{n-1} + 1)P$
**for** $i = n-2, \cdots, 0$ **do**
  **if** $k_i = 0$ **then**
    $R_2 = 3R_0$, $R_1 = 2R_0 + R_1$
  **end if**
  **if** $k_i = 1$ **then**
    $R_2 = 2R_0 + R_1$, $R_1 = 2R_1 + R_0$
  **end if**
  **if** $k_i = 2$ **then**
    $R_2 = 2R_1 + R_0$, $R_1 = 3R_1$
  **end if**
  $R_0 = R_2$
**end for**
Return $Q = R_0$
End

**Example 2.** $k = 520$, $k$ ={2,0,1,0,2,1}
$R_0 = 2P, R_1 = 3P$
$k = 0, R_0 = 3R_0 = 6P, R_1 = 2R_0 + R_1 = 7P$
$k = 1, R_0 = 2R_0 + R_1 = 19P, R_1 = 2R_1 + 2R_0 = 20P$
$k = 0, R_0 = 3R_0 = 57P, R_1 = 2R_0 + R_1 = 58P$
$k = 2, R_0 = 2R_1 + R_0 = 173P, R_1 = 3R_1 = 174P$
$k = 1, R_0 = 2R_0 + R_1 = 520P, R_1 = 2R_1 + 2R_0 = 521P$
**Output** $Q = 520P$

and the variable $R_1$ is updated as

$$R_1 = \begin{cases} 2R_0 + R_1, & \text{if } k_i = 0 \\ 2R_1 + R_0, & \text{if } k_i = 1 \\ 3R_1, & \text{if } k_i = 2 \end{cases} \quad (7)$$

The new proposed algorithm preserves the advantages of the original Montgomery ladder algorithm, the difference between input point $R_0$ and $R_1$ is not changed, *i.e.*, $R_1 - R_0 = P$.

## 3.2 Composite Operation

Computing $2P_1 + P_2$. Let O is the identity element on the elliptic curve, which is considered as a point at infinity.

Now given two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ in E\{O} with $x_1 \neq x_2$, their sum is the point $R = P_1 + P_2 = (x_3, y_3)$ and is given by using Equation (3)

$$x_3 = \mu_1^2 - a - x_1 - x_2, y_3 = \mu_1(x_1 - x_3) - y_1, \quad (8)$$

where $\mu_1 = \dfrac{y_2 - y_1}{x_2 - x_1}$.

R is added to $P_1$ to get point $S = 2P_1 + P_2 = (x_4, y_4)$, and coordinates of $S$ is given by

$$x_4 = \mu_2^2 - a - x_1 - x_3, y_4 = (x_1 - x_4)\mu_2 - y_1, \quad (9)$$

where $\mu_2 = \dfrac{y_3 - y_1}{x_3 - x_1}$.

The calculation of $y_3$ can be omitted by deform $\mu_2$ as

$$\mu_2 = -\mu_1 - \frac{2y_1}{x_3 - x_1} = -\frac{y_1}{x_1 - x_3} - \mu_1.$$

$x_4$ can be also computed as

$$x_4 = \mu_2^2 - a - x_1 - x_3 = (\mu_2 - \mu_1)(\mu_2 + \mu_1) + x_2.$$

In addition, letting $h := (x_2 - x_1)^2(2x_1 + x_2) - (y_2 - y_1)^2 + a(x_2 - x_1)^2$,it follows that $h = (x_2 - x_1)^2(x_1 - x_3)$. Defining $H := h(x_2 - x_1)$ and $I := H^{-1}$,we get

$$\frac{1}{x_2 - x_1} = hI \text{ and } \frac{1}{\mathrm{x_1 - x_3}} = (\mathrm{x_2 - x_1})^3\mathrm{I}.$$

Therefore, there is no $x_3$ is used when computing $2P_1 + P_2$. In Algorithm 5, the computation of $h, H, I, \mu_1$ and $\mu_2$ requires 1 inversion, 2 squarings, 1 cubing and 8 multiplications. Similarly, Algorithm 6 is point tripling algorithm.

**Algorithm 5** Double-and-Add Algorithm for Elliptic Curve over GF($3^m$)

**Input:** $P_1 = (x_1, y_1) \neq \mathrm{O}$, and $P_2 = (x_2, y_2) \neq \mathrm{O}$
**Output:** $S = 2P_1 + P_2$.
**if** $x_1 = x_2$ **then**
  **if** $y_1 = y_2$ **then**
    return $3P_1$
  **end if**
  **if** $y_1 \neq y_2$ **then**
    return $P_1$
  **end if**
  $X \leftarrow (x_2 - x_1)^2; Y \leftarrow (y_2 - y_1)^2$
  $h \leftarrow X(2x_1 + x_2) - Y + aX$
  **if** $h$=0 **then**
    return O
  **end if**
  $H \leftarrow h(x_2 - x_1); I \leftarrow H^{-1}$
  $\mu_1 \leftarrow hI(y_2 - y_1)$
  $\mu_2 \leftarrow -y_1X(x_2 - x_1)I - \mu_1$
  $x_4 \leftarrow (\mu_2 - \mu_1)(\mu_2 + \mu_1) + x_2$
  $y_4 \leftarrow (x_1 - x_4)\mu_2 - y_1$
**end if**
Return $(x_4, y_4)$
End

# 4 Analysis of Algorithm

In this part, we first analyze the security of the new algorithm. Then, in order to make the efficiency analysis more accurate, we compare the new algorithm with previous algorithms over GF($3^m$) in different coordinate systems. And some practical results are listed in the table below.

**Algorithm 6** Tripling Algorithm for Elliptic Curve over GF($3^m$)

**Input:** $P_1 = (x_1, y_1) \neq \mathrm{O}$
**Output:** $S = 3P_1$.
**if** $y_1 = 0$ **then**
  return $P_1$
**end if**
$A \leftarrow ax_1; B \leftarrow x_1^3 + b$
$C_0 \leftarrow 1; C_1 \leftarrow a(x_1^3 + b) = aB$
$D \leftarrow y_1^3$
$E \leftarrow B^3 - bA^3C_0^2$
$F \leftarrow D^3 - aDC_1^2$
$x_3 \leftarrow \frac{E}{C_1^2}$
$y_3 \leftarrow \frac{F}{C_1^3}$
Return $(x_3, y_3)$
End

## 4.1 Security Analysis

The basic idea of power analysis attack is to obtain its key by analyzing the energy consumption during the operation of the cryptographic device. Power analysis attacks include Simple Power Analysis (SPA) and Differential Power Analysis (DPA). SPA [10,14,23] is a technique that can directly analyze the power consumption information, which is collected during the execution of the encryption algorithm. It can retrieve its key through a single leakage trace.

In this paper, the security is analyzed by taking $I/M = 8.75, C/M = 1.37, S/M = 1$ [27,29] for the cost of inverse, cubing and squaring operation. For GF($3^m$), the cost of computing $2P_1 + P_2$ is $1I + 2S + C + 8M \approx 1I + 11M$. And the cost of computing $3P_1$ is $1I + 2S + 6C + 3M \approx 1I + 11M$. Because energy consumption of each iteration is the same roughly whether $k_i = 0, 1,$ or 2, so side channel profile of $2P_1 + P_2$ and $3P_1$ can not be distinguished. Our algorithm is based on the original Montgomery ladder algorithm, so the this algorithm is able to resist SPA attacks. Furthermore, this algorithm can resist the DPA by randomizing the scalar.

## 4.2 Efficiency Analysis

Before we analyze the efficiency, we define $\beta$ as a dynamic ratio of inverse and multiplication [13]:

$$\frac{I}{M} = \beta. \tag{10}$$

In Algorithm 4, there are operation included tripling and double-and-add or two double-and-add at each loop, which depends on the value of the scalar $k$. The cost of tripling and double-and-add is $2I + 4S + 7C + 11M$, the cost of two double-and-double is $2I + 4S + 2C + 16M$, so the average cost of each iteration, which is tripling and double-and-add or two double-and-add, is $2I + \frac{38}{3}M + \frac{16}{3}C + 4S$.

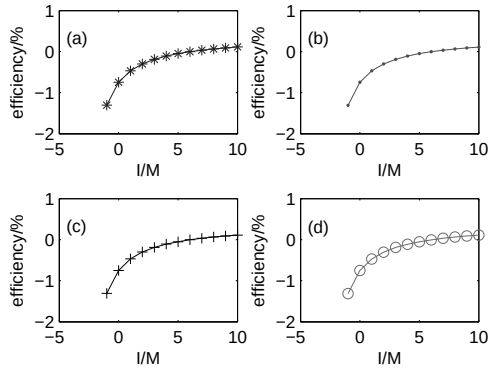


Figure 1: The comparison between Algorithm 3 and Algorithm 4 in the (a) m=101, (b) m=122, (c) m=162, (d) m=379

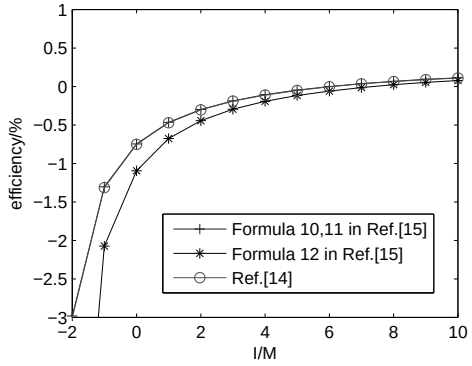


Figure 2: The comparison of algorithms with the scalar of the equivalent bits

#### 4.2.1 Affine Coordinate

The proposed Algorithm 4 is compared with previous algorithm in affine coordinate system. An analysis of the costs of different scalar multiplication is shown in Table 1.

Given an integer $k$, the ternary (3-adic) expansion of $k$ is shorter than the binary (2-adic) expansion. Suppose $k$ is an $n$ bit number, and $n = \lceil \log_2 k \rceil$, the length of the ternary expansion of $k$ is $m$, and $m = \lceil \log_3 k \rceil$, therefore, $n = m \log_2 3 \approx 1.584m$, *i.e.*, 101-ternary is equivalent to 160-binary [27], 122, 162, 379-ternary is equivalent to 192, 256, 600-binary, respectively.

For example, comparing Algorithm 3 using formula 10, 11 in [29] and our algorithm, we can conclude that the efficiency of the new algorithm is recorded in the following formula:

$$\varepsilon = 1 - \frac{(2m-2)\beta + (\frac{50m}{3} - \frac{50}{3})}{(2mlog_2 3 - 2)\beta + (6mlog_2 3 - 6)}. \qquad (11)$$

Figure 1 (a)-(d) show that the comparison between the original Montgomery ladder algorithm, *i.e.*, Algorithm 3, and the improved ternary Montgomery ladder algorithm, *i.e.*, Algorithm 4, in different data bits. Our new algorithm can improve the computational efficiency of scalar multiplication with the increase of $\beta$. Compared with Algorithm 3 using formula 10, 11 in [29], the average efficiency of the new algorithm can be increased by 6.6% and 11.3% for different data bits, respectively, when $\beta$ is equal to 8 and 10.

Figure 2 shows the comparison of algorithms with the scalar of the equivalent bits. From the graph of the variation of improved efficiency with the ratio of inverse and multiplication, we can conclude that the larger the ratio of $I$ and $M$, the slower the rate of increasing in efficiency, and Table 1 shows that the efficiency of the new Montgomery ladder algorithm is increased by 7.8% and 11.0%, compared to Algorithm 3 using formula 12 in [29] and [4], when $\beta$ is equal to 10.

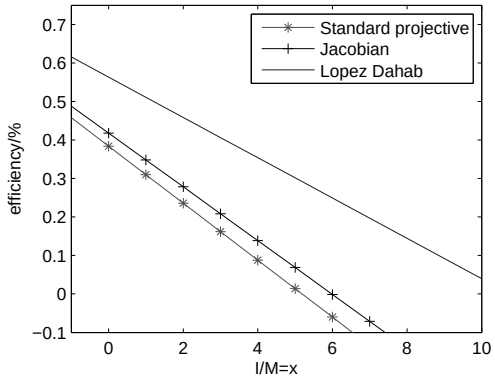


Figure 3: The comparison of algorithms in different coordinates in [27]

#### 4.2.2 Projective Coordinate

As is shown in Table 2, the proposed Algorithm 4 is compared with previous algorithm, such as [27] and [30], in projective coordinate system. The peculiarity of this article is the dynamic ratio $\beta$, resulting in a dynamic percentage. In this paper, we take a list of the efficiency at special points.

Figure 3 shows the comparison of algorithms in different projective coordinates in [27], we can draw that the efficiency decreases, as the ratio increases. For algorithm in Lopez Dahab coordinate, it has a slow rate of decline, compared with the other two projective coordinates. Form Table 2, we draw that if we set the maximum value of the ratio as 10, the efficiency is improved by 4.0%.

Figure 4 shows the comparison of algorithms in different projective coordinates in [30]. Compared with the other two previous algorithm, the algorithm in Co-Z projective has higher efficiency. In addition, in Table 2, compared our algorithm, previous algorithm is proposed by 5.0% in standard projective, when $\beta$ is equal to 3, and 7.2% in scaled projective [3], when $\beta$ is equal to 2, and 4.9% in Co-Z projective, when $\beta$ is equal to 1.5, respectively.

Table 1: Timing costs of different algorithm in affine coordinate system

| **Algorithm** | **Total costs**($n = \lceil \log_2 \boldsymbol{k} \rceil, m = \lceil \log_3 \boldsymbol{k} \rceil$) | **Total costs**($\#I + \#M$) |
|---|---|---|
| Formula 10,11 in [29] | $2(n-1)I + 4(n-1)M + 2(n-1)S$ | $(2n-2)I + (6n-6)M$ |
| Formula 12 in [29] | $2(n-1)I + 3(n-1)M + 2(n-1)C + 2(n-1)S$ | $(2n-2)I + (5n-5)M$ |
| [4] | $(2n-3)I + (3n-5)M + (3n-5)S$ | $(2n-3)I + (6n-10)M$ |
| Ours | $2(m-1)I + \frac{38}{3}(m-1)M + \frac{16}{3}(m-1)C + 4(m-1)S$ | $(2m-2)I + \frac{50}{3}(m-1)M$ |

Table 2: Timing costs of different algorithm in projective coordinate system

| **Algorithm** | **Coordinate** | **Each iteration's costs** | **Total costs**($\#M$) | $\frac{I}{M} = \beta$ | $\varepsilon$ |
|---|---|---|---|---|---|
| Ref. [27](binary) | Standard projective | $15M + 2S + 4C$ | $17M$ | 4 | 8.8% |
| | Jacobian | $13M + 5S + 5C$ | $18M$ | 5 | 6.9% |
| | Lopez Dahab | $17M + 7S + 2C$ | $24M$ | 8 | 17.9% |
| | | | | 10 | 4.0% |
| Ref. [30](binary) | Standard projective | $13M + 2S + 2C$ | $15M$ | 3 | 5.0% |
| | Scaled projective [3] | $14M + 3C$ | $14M$ | 2 | 7.2% |
| | Co-Z projective | $10M + 3S + C$ | $13M$ | 1.5 | 4.9% |
| Ours(ternary) | Affine | $2I + \frac{38M}{3} + \frac{16C}{3} + 4S$ | $2I + \frac{50M}{3}$ | | |

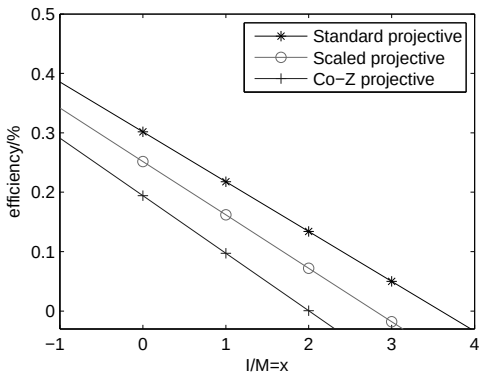


Figure 4: The comparison of algorithms in different coordinates in [30]

In the actual experimental environment, I/M is different and relatively large. Table 2 shows that the efficiency of new algorithm varies from 4% to 17.9%, compared with previous algorithm in projective coordinate system.

## 5 Conclusions

In this paper, we proposed an improved Montgomery ladder algorithm over GF($3^m$), and the scalar was expressed in ternary form. In addition, we derived composite operation formulas $2P_1 + P_2$ and $3P_1$ with a lower computational cost. Based on the original Montgomery ladder algorithm, it is able to resist SPA. In analyzing efficiency, the difference with the past is that the ratio of $I$ to $M$, which is set to a dynamic value. Correspondingly, increased efficiency is also dynamic, ranging from 7.8% to 11.3% in affine coordinate, and from 4% to 17.9% in projective coordinate. Further work may include designing a new scalar multiplication algorithm over GF($3^m$), and it makes cubing operation faster.

## References

[1] N. Christophe and J. M. Robert, "New parallel approaches for scalar multiplication in elliptic curve over fields of small characteristic," *IEEE Transactions on Computers*, vol. 64, no. 10, pp. 2875–2890, 2015.

[2] M. Ciet, M. Joye, K. Lauter, and P. L. Montgomery, "Trading inversions for multiplications in elliptic curve cryptography," *Designs, Codes and Cryptography*, vol. 39, no. 2, pp. 189–206, 2006.

[3] R. R. Farashahi, H. F. Wu, and C. A. Zhao, "Efficient arithmetic on elliptic curves over fields of characteristic three," in *International Conference on Selected Areas in Cryptography*, pp. 135–148, Aug. 2012.

[4] H. H. Gu, W. L. Xie, and Ray C. C. Cheung, "Analysis the montgomery ladder algorithm for elliptic curves over ternary fields," in *International Conference on Information and Network Security (ICINS'13)*, pp. 1–5, Nov. 2013.

[5] L. D. Han, Q. Xie, and W. H. Liu, "An improved biometric based authentication scheme with user anonymity using elliptic curve cryptosystem," *International Journal of Network Security*, vol. 19, no. 3, pp. 469–478, 2017.

[6] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag New York, 2004. (`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.394.3037&rep=rep1&type=pdf`)

[7] G. F. Hou and Z. J. Wang, "A robust and efficient remote authentication scheme from elliptic curve cryptosystem," *International Journal of Network Security*, vol. 19, no. 6, pp. 904–911, 2017.
[8] C. Kakali, D. Asok, and Daya Gupta, "Mutual authentication protocol using hyperelliptic curve cryptosystem in constrained devices," *International Journal of Network Security*, vol. 15, no. 1, pp. 9–15, 2013.
[9] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.
[10] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Lecture Notes in Computing Science Springer-Verlag*, vol. 1666, pp. 388–397, 1999.
[11] K. Koyama and Y. Tsureoka, "Speeding up elliptic curve cryptosystems by using a signed binary windows method," *Advances in Cryptology (CRYPTO'92)*, pp. 345–357, 1992.
[12] H. K. Kwang, I. K. So, and S. C. Ju, *New Fast Algorithms for Arithmetic on Elliptic Curves Over Finite Fields of Characteristic Three*, May 2007. (`http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.81.1113`)
[13] L. Li, "Research on the ternary algorithm in the elliptic curve operations," *Journal of Network Safety Technology and Application (in Chinese)*, no. 11, pp. 94–96, 2015.
[14] S. G. Liu, H. T. Yao, and X. A. Wang, "Spa resistant scalar multiplication based on addition and tripling indistinguishable on elliptic curve cryptosystem," in *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC'15)*, pp. 785–790, Nov. 2015.
[15] J. Lopez and R. Dahab, "Fast multiplication on elliptic curves over gf(2m) without precomputation," in *First International Workshop on Cryptographic Hardware and Embedded Systems (CHES'99)*, pp. 316–327, Aug. 1999.
[16] V. S. Miller, "Use of elliptic curves in cryptography," *Lecture Notes in Computer Science Springer-Verlag*, vol. 218, no. 1, pp. 417–426, 1986.
[17] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Mathematics of Computation*, vol. 48, no. 177, pp. 243–264, 1987.
[18] J. Moon, D. Lee, and J. Jung, "Improvement of efficient and secure smart card based password authentication scheme," *International Journal of Network Security*, vol. 19, no. 6, pp. 1053–1061, 2017.
[19] C. Negre, "Scalar multiplication on elliptic curves defined over fields of small odd characteristic," in *Proceedings of the 6th International Conference on Cryptology (INDOCRYPT'05)*, pp. 389–402, Dec. 2005.
[20] K. Okeya and T. Takagi, "The width-w naf method provides small memory and fast elliptic saclar multiplications secure against side channel attacks," *Rsa Conference on the Cryptographers*, pp. 328–343, 2003.
[21] T. Oliveira, J. López, and R. H. Francisco, "The montgomery ladder on binary elliptic curves," *Journal of Cryptographic Engineering*, no. 5, pp. 1–18, 2017.
[22] Q. Qian, Y. L. Jia, and R. Zhang, "A lightweight rfid security protocol based on elliptic curve cryptography," *International Journal of Network Security*, vol. 18, no. 2, pp. 354–361, 2016.
[23] Reddy and E. Kesavulu, "Elliptic curve cryptosystems and side-channel attacks," *International Journal of Network Security*, vol. 12, no. 3, pp. 151–158, 2011.
[24] T. Satoh, "The canonical lift of an ordinary elliptic curve over a finite field and its point counting," *Journal of the Ramanujan Mathematical Society*, vol. 15, no. 4, pp. 247–270, 2000.
[25] S. F. Shen and M. Zhou, "Research on fast algorithms for scalar multiplication of elliptic curve cryptography over gf(3n)," *Advances in Applied Mathematics (in Chinese)*, vol. 4, no. 4, pp. 390–399, 2015.
[26] C. S. Sin, *Regular Ternary Algorithm for Scalar Multiplication on Elliptic Curves Over Finite Fields of Characteristic Three*, July 2012. (`https://eprint.iacr.org/2012/390.pdf`)
[27] N. P. Smart and E. J. Westwood, "Point multiplication on ordinary elliptic curves over fields of characteristic three," *Applicable Algebra in Engineering, Communication and Computing*, vol. 13, no. 6, pp. 485–497, 2003.
[28] S. F. Tzeng, M. S. Hwang, "Digital signature with message recovery and its variants based on elliptic curve discrete logarithm problem", *Computer Standards & Interfaces*, vol. 26, no. 2, pp. 61–71, Mar. 2004.
[29] H. Wang, B. Li, and W. Yu, "Montgomery algorithm on elliptic curves over finite fields of character three," *Journal on Communications (in Chinese)*, vol. 29, no. 10, pp. 25–29, 2008.
[30] W. Yu, B. LI, K. P. Wang, W. X. Li, and S. Tian, "Co-z montgomery algorithm on elliptic curves over finite fields of characteristic three," *Chinese Journal of Computers (in Chinese)*, vol. 40, no. 5, pp. 1121–1133, 2017.

## Biography

**Shuang-Gen Liu** is PhD, associate professor. His research interests are information security and cryptology.

**Rong-Rong Wang** is a graduate student of Xi'an University of posts and telecommunications. She is mainly engaged in the research of elliptic curve cryptosystem.

**Yun-Qi Li** is a undergraduate student in information security in Xi'an University of posts and telecommunications. Her research interest is cryptography.

**Can-Liang Zhai** is a undergraduate student in information security in Xi'an University of posts and telecommunications. His research interest is cryptography.