2017

Efficient Computation of Miller's Algorithm in Pairing-Based Cryptography

Shun Wang
University of Windsor

Efficient Computation of Miller's Algorithm in Pairing-Based Cryptography

by

Shun Wang

A Thesis

Submitted to the Faculty of Graduate Studies

through Electrical and Computer Engineering

in Partial Fulfillment of the Requirements for

the Degree of Master of Applied Science at the

University of Windsor

Windsor, Ontario, Canada

2017

Efficient Computation of Miller's Algorithm in Pairing-Based Cryptography

by

Shun Wang

APPROVED BY:

J. Lu
School of Computer Science

M. Mirhassani
Department of Electrical and Computer Engineering

H. Wu, Advisor
Department of Electrical and Computer Engineering

May 18, 2017

AUTHOR'S DECLARATION OF ORIGINALITY

ABSTRACT

Pairing-based cryptography (PBC) provides novel security services, such as identity-based encryption, attribute-based encryption and anonymous authentication. The Miller's Algorithm is considered one of the most important algorithms in PBC and carries the most computation in PBC.

In this thesis, two modified Miller's algorithms are proposed. The first proposed algorithm introduces a right-to-left version algorithm compared to the fact that the original Miller's algorithm works only in the fashion of left-to-right. Furthermore, this new algorithm introduces parallelable computation within each loop and thus it can achieve a much higher speed. The second proposal has the advantage over the original Miller's algorithm not only in parallelable computation but also in resistance to certain side channel attacks based on the new feature of the equilibrium of computational complexities.

An elaborate comparison among the existing works and the proposed works is demonstrated. It is expected that the first proposed algorithm can replace the original Miller's if a right-to-left input style is required and/or high speed is of importance. The second proposed algorithm should be chosen over the original Miller's if side channel attack is a concern.

DEDICATION

To my adorable wife, my loving parents, and my respectful parents in-law:

Wife: Xi Chen

Father: Zhongyan Wang

Mother: Yaping Yu

Father in-law: Baofeng Chen

Mother in-law: Yan Li

ACKNOWLEDGMENTS

I would like to express my faithful gratitude to everyone who helped me. First of all, I appreciate my wife's deep love and full support, as well as the encouragement and financial support from my parents and my parents in-law. Without them, I could not overcome all difficulties and accomplish my study.

Furthermore, I am quite grateful to my supervisor Dr. Huapeng Wu, the Professor of Electrical and Computer Engineering at University of Windsor. He has instructed me throughout my research and this thesis. As one of best teachers I have ever had, Dr. Wu impressed upon me that a brilliant teacher edifies students in matters far beyond those in books and academy. His extensive knowledge and logical thinking are invaluable; without his elaborate and constructive comments on my research, this thesis could be impossible.

I thank my friends, Siyu Zhang, Ruiqing Dong, Bingxin Liu, Chen Chen and Yue Huang. They gave me their help and time during the adversity of my study.

Ultimately, I hope to show my appreciation to the faculties of Electrical and Computer Engineering at University of Windsor since their efforts during my study for the master degree. Furthermore, I pretty appreciate the financial support from the University of Windsor and my supervisor Dr. Huapeng Wu.

Shun Wang

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ACRONYMS

$\#E$	number of points on E
$\mathbb{F}_q$	finite field with prime number q elements
$\mathbb{F}_{q^k}$	full extension field
$\mathbb{G}$	a group of the bilinear map
D	a divisor
$Deg(D)$	the degree of a divisor D
e	bilinear map
$E(\mathbb{F}_q)$	elliptic curve over a finite field with prime number q elements
$E(\mathbb{F}_{q^k})$	elliptic curve over a full extension field
r	the largest prime order of a group in $E(\mathbb{F}_q)$
$supp(D)$	the support of D
CPPA	Conditional Privacy-preserving Authentication
DSRC	Dedicated Short Range Communication
ECC	Elliptic Curve Cryptosystem
PBC	Pairing-based Cryptography

1 INTRODUCTION

1.1 Pairing-based Cryptography and Its Applications

The Internet becomes increasingly important in our modern society. The Internet technology has also progressed at a constant step to provide new and better services to meet the demands from its users. Pairing-based cryptography (PBC) is an emerging research area in the field of cryptography [4], which provides several new cryptographic services over the Internet complement to conventional symmetrical and public key cryptosystems. Some features and important facts about PBC include:

- PBC can provide several special security services, i.e., identity-based encryption, attribute-based encryption and anonymous authentication, which are not readily available from the conventional symmetrical and public key cryptosystems.

- PBC studies mathematical bilinear function that can map a very complex computational problem to a relatively simple one without compromising its security strength.

- Pairing-based cryptography technology has been recently standardized in 2013 in P1363.3 "IEEE Standard for Identity-Based Cryptographic Techniques using Pairings" [5].

Pairing-based cryptography can provide many unique or more efficient cryptography and security services for the Internet, compared to conventional cryptographic technology [6]. Its important applications are introduced as follows,

- Identity-based encryption [7]: in public key encryption system, the public key of any user is based on his own identity. PBC is able to construct new ID-based cryptographic primitives [8] to complement the conventional public key cryptosystems.

- Key exchange: PBC can make a tripartite key exchange be done in one round [9].
- Short signatures: Boneh-Lynn-Shacham (BLS) signature schemes [10] of PBC only use a half of the length of other signature schemes [5].
- Anonymous authentication: the research work [11] has shown that pairing-based cryptography can be applied to vehicular standard Dedicated Short Range Communication (DSRC) [12]:
  - DSRC is a communication service to distribute a message from a vehicle to all other vehicles or infrastructures to overcome the problem of the high mobility environment [13] [14].
  - The goals of DSRC are increasing road capacity [15], avoiding accidents, providing web or entertainment services. [16]
    - The Conditional Privacy-preserving Authentication (CPPA) technique [17] is one kind feasible scheme for DSRC, and it's defined by the following algorithms: system setup, key generation, anonymous authentication, and conditional tracking. These algorithms all use pairing-based cryptography.

Bilinear map plays a central role in pairing-based cryptography. The popular implementations of bilinear map are Weil pairing [18] and Tate pairing [19]. Miller's Algorithm [20], which is used to compute the Weil pairing and Tate pairing, is probably the most important and most computation-intensive algorithm in pairing-based cryptography. This thesis proposes novel research works on improvement to Miller's Algorithm with computational efficiency and enhanced security.

1.2 Research Contribution

This thesis work concentrates on computational efficiency and security strength of pairing-based cryptography. Since Miller's Algorithm [20] is considered as the core algorithm in pairing-based cryptography and most computational intensive, our proposed work is aiming to improvement to Miller's Algorithm [21] in terms of its computational efficiency and resistance to side channel attacks. The proposed work can be summarized as follows.

The original Miller's Algorithm works in a manner of left to right. In this thesis a right to left (R2L) version for Miller's Algorithm is proposed. Moreover, the new R2L algorithm has the feature of parallelism while the original version does not have. When the algorithm is implemented in parallel architecture, it can be expected that the proposed algorithm is much faster than the original Miller's.

The second proposed work is a modified Miller's Algorithm with enhanced security. Compared to Miller's Algorithm, the proposed algorithm not only makes parallel computation possible but also has the nice property of resistance to certain side channel attacks, i.e., simple power analysis.

The idea of using signed-digit binary number representation in Miller's Algorithm was first discussed in [3]. As an addition to the proposed works, an error in the algorithm presented in [3] is found and corrected in this thesis.

1.3 The Scope and Organization of the Thesis

The organization of the rest of this thesis is as follows. In Chapter 2, mathematical fundamentals which contain the modular operations, groups, finite fields and elliptic curves over a finite field are introduced. In Chapter 3, the divisors and bilinear map are explained, which provides important theoretical and algorithmic basis for comprehending pairing-based cryptography. In Chapter 4 of the thesis, Weil pairing and Miller's Algorithm are summarized. Subsequent works on Miller's Algorithm are also reviewed and explained. The New Right-to-left Miller's Algorithm and the Modified Miller's Algorithm with Enhanced Security are proposed in Chapter 5. In Chapter 6, the complexities of the existing works and the proposed works are analyzed and compared. It has been shown that the proposed algorithms have clear advantages to the original Miller's or its version using signed-digit binary number, in terms of parallel-able computation and resistance to certain side channel attacks. Finally, the conclusion and possible future work are discussed in Chapter 7.

2 MATHEMATICAL PRELIMINARIES

Finite field and elliptic curve are the cornerstones of pairing-based cryptography. In this chapter, we introduce fundamental concepts, such like groups, finite fields and their arithmetic, as well as elliptic curve defined over a finite field, and elliptic curve point operations.

2.1 Modular Operations

2.1.1 Modular Operations (Integer)

1. x mod n means "the remainder of n dividing x" [22]. In other words, if $x = an + b$, and $a, b \in integer$ as well as $0 \leq b \leq n - 1$, then x mod $n = b$.

2. Inverse: If $ax = 1$ mod n, then a is the inverse of x mod n [22]. There are two popular methods to solve a:

   - *Method 1:* Try every value for $a < n$ until xa mod $n = 1$.
   - *Method 2:* Euclidean method, which is usually used to solve the inverse of big integers, so it is recommended to use Method 1 to solve the inverse of small integers. No matter what usage Euclidean method is, Table 2.1 just demonstrates how Euclidean method works with the $5a$ mod $7 = 1$.

Table 2.1: Euclidean Method to Solve Inverse

Step i	b	a	d	k	Equality
0	1	0	7		$1 \times 7 + 0 \times 5 = 7$
1	0	1	5	1	$0 \times 7 + 1 \times 5 = 5$
2	1	-1	2	2	$1 \times 7 + (-1) \times 5 = 2$
3	-2	3	1		$(-2) \times 7 + 3 \times 5 = 1$

The followings are the explanation of Table 2.1.

- $b_0 = 1, a_0 = 0, b_1 = 0,$ and $a_1 = 1$ are fixed, as well as k_0 is null;
- $d_0 = 7, d_1 = 5$ are given, then k_1 is the quotient of d_1 dividing d_0;
- $b_2 = b_0 - b_1 k_1$;
- $a_2 = a_0 - a_1 k_1$;
- $d_2 = d_0 - d_1 k_1$, d_2 is also the remainder of d_1 dividing d_0;
- k_2 is the quotient of d_2 dividing d_1;
- Similarly, $b_3 = b_1 - b_2 k_2$, $a_3 = a_1 - a_2 k_2$, $d_3 = d_1 - d_2 k_2$, k_3 is the quotient of d_3 dividing d_2;

  ...

  $b_i = b_{i-2} - b_{i-1} k_{i-1}$, $a_i = a_{i-2} - a_{i-1} k_{i-1}$, $d_i = d_{i-2} - d_{i-1} k_{i-1}$, k_i is the quotient of d_i dividing d_{i-1}.
- Until $d_i = 1$ is gotten, stopping calculating and the value of a_i is the final answer. Additionally, k_i is unnecessary to be computed.
- In this instance, $d_3 = 1$, so $a_3 = 3$ is the answer.

2.1.2 Modular Operations (Polynomial)

1. Definition: $f(x)$ mod $P(x)$ means "the remainder of $(f(x) \div P(x))$" [22].

   - It can be denoted $f(x) = a(x)P(x) + b(x)$, where the degree of $b(x)$ is lower than that of $P(x)$, then $f(x)$ mod $P(x) = b(x)$.
   - Polynomial division: $(f(x) \div P(x))$ to reap the remainder.

2. For example: $x^8 + 1$ mod $x^3 + x^2 + 1 = 6x^2 - 3x + 5$, and the quotient is $x^5 - x^4 + x^3 - 2x^2 + 3x - 4$

2.2 Groups

2.2.1 Definition

A group is a set G together with a binary operation $*$ on G such that:

1. Binary operator $*$ is associative, i.e., for any $a, b, c \in G$,
$$a * (b * c) = (a * b) * c.$$
2. There exists an identity (or unity) element $e \in G$, i.e., for all $a \in G$,
$$a * e = e * a = a.$$
3. For each $a \in G$, there an inverse element $a^{-1} \in G$, such that
$$a * a^{-1} = a^{-1} * a = e.$$

2.2.2 Types of Groups

1. The operation $*$ can be like ordinary multiplication or addition:
    - Multiplicative group (and the unity is $e = 1$, if $*$ is multiplication.)
        - Usually denoted by $G^{\times}$.
    - Additive group (and the identity is $e = 0$, if $*$ is addition.)
        - Usually denoted by G^{+}.
2. Infinite groups and finite groups
    - Infinite group: there are infinite many elements in a group.
    - Finite group: there are finite many elements in a group.

2.2.3 The Important Concepts of Groups

The computation details of the following concepts will be exemplified in "2.3 Background Knowledge of Finite Field" . Hence, here just shows the outcomes.

1. A group G is called cyclic group if there exists a group element g such that any other element in G can be written as g^j for a certain integer $j > 1$.
   - In this case, the group element g is called a generator of G, or a primitive element in G.
   - Example: $G^{\times} = \{1, 2, 3, 4\}$ under mod-5 multiplication is a cyclic group with a primitive element 2. Because all the other group elements can be written as a power of 2: $G^{\times} = \{1, 2, 3, 4\} = \{2^4, 2^1, 2^3, 2^2\}$.
2. The order of group element a is defined as the minimal positive integer i such that $a^i = \textit{the unity}$ (or $a^i = 1$ since the unity is 1 in this case). It is written $ord(a) = i$. Clearly, a primitive element has the maximal order.
   - In group $G^{\times} = \{1, 2, 3, 4\}$ under mod-5 multiplication, $ord(1) = 1$; $ord(2) = 4$; $ord(3) = 4$; $ord(4) = 2$.
   - $G^{\times} = \{1, 2, 3, ..., p-1\}$ under mod-p multiplication is a cyclic group, where p is a prime.
     - A primitive element in $G^{\times}$ has the maximal order of $p - 1$.
     - Any other possible order of an element in this group has to be a factor of $p - 1$.
     - For k being a factor of $p - 1$, there always exists an element in the group that has order of k.

2.3 Finite Fields

2.3.1 Definition

Finite field (or Galois field) is a set that has finitely many elements, and the result, which is operated by addition and multiplication of any two elements, is still closed in the same set [22].

Note that the "closed" means the result, which is computed by any two elements, still belongs to the same set, namely, the same finite field.

In other words to explain the definition, it is a set of finite many elements where addition and multiplication are defined [23].

- The finite field is an additive group under the addition operation.
- All the nonzero elements in a finite field form a multiplicative group under multiplication operation.

There are several popular families in finite fields, ($\mathbb{F}$ is used to denote "Finite Field"), such as $\mathbb{F}_q$, $\mathbb{F}_{2^k}$, $\mathbb{F}_{3^k}$, and $\mathbb{F}_{q^k}$ [24]. Whereas, in this thesis, it just concerns and discusses the $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$ [25].

1. **Finite field $\mathbb{F}_q$, where q is a prime number:**
    - The set is written as: $\mathbb{F}_q = \{0, 1, 2, 3, ..., q-1\}$.
    - The operations: mod q addition, or mod q multiplication.
2. **Finite field $\mathbb{F}_{q^k}$, where q is a prime number, and k is an integer > 1:**
    - The set is written as: $\mathbb{F}_{q^k} = \{$polynomials of degree up to $k-1$ with coefficients belonging to $\mathbb{F}_q = \{0, 1, 2, 3, ..., q-1\}$, with irreducible polynomial $f(x)$.$\}$
    - The operations: mod q polynomial addition; mod $f(x)$ and mod q polynomial multiplication.

2.3.2 The Arithmetic of Finite Field

It is an easy understanding way that demonstrates the arithmetic of finite field with some examples of specific numbers operations.

1. Finite field $\mathbb{F}_5$, and $\mathbb{F}_5 = \{0, 1, 2, 3, 4\}$:

   - *Mod q addition:* $a + b = (a + b) \bmod q$. eg: $(3 + 4) \bmod 5 = 2$; $(2 + 2) \bmod 5 = 4$.
   - *Mod q multiplication:* $a \times b = (a \times b) \bmod q$. eg: $(3 \times 3) \bmod 5 = 4$; $(2 \times 4) \bmod 5 = 3$.

2. Finite field $\mathbb{F}_{2^2}$, and $\mathbb{F}_{2^2} = \{0, 1, x, x + 1\}$ with irreducible polynomial $f(x) = x^2 + x + 1$:

   - Irreducible polynomials:
     - Irreducible polynomial is similar to prime number for integers, an irreducible polynomial of degree n does not have a factor polynomial of degree between 1 and $n - 1$.
     - Table 2.2 is a list of irreducible polynomials over $\mathbb{Z}_2 = \{0, 1\}$.

Table 2.2: A List of Irreducible Polynomials over $\mathbb{Z}_2 = \{0, 1\}$

n	irreducible polynomial $f(x)$, (only one listed for each n)
2	$x^2 + x + 1$
3	$x^3 + x + 1$
4	$x^4 + x + 1$
5	$x^5 + x^2 + 1$
6	$x^6 + x + 1$
7	$x^7 + x + 1$
8	$x^8 + x^4 + x^3 + x + 1$

- *Addition:* $a + b = (a + b) \bmod 2$. For example,
  - $(x + (x + 1)) \bmod 2 = (2x + 1) \bmod 2 = 1$
  - $(1 + (x + 1)) \bmod 2 = (x + 2) \bmod 2 = x$
- *Multiplication:* $a \times b = (a \times b) \bmod f(x) \bmod 2$. For example,
  - $(x \times (x+1)) \bmod f(x) \bmod 2 = (x^2 + x) \bmod (x^2 + x + 1) \bmod 2 = 1$
  - $((x + 1) \times (x + 1)) \bmod f(x) \bmod 2 = (x^2 + 2x + 1) \bmod (x^2 + x + 1) \bmod 2 = x$

3. Another representation $\mathbb{F}_{2^2}$, and $\mathbb{F}_{2^2} = \{0x+0, 0x+1, x+0, x+1\} = \{00, 01, 10, 11\}$: the operations of addition and multiplication are similar to the previous representation.

2.3.3 The Order of a Finite Field Element

1. For any $a \neq 0$ and $a \in \mathbb{F}_q$, the minimal positive integer j for $a^j = 1$ is called the order of a, denoted by $ord(a)$.

2. a^i with $i = 1, 2, ..., j-1$ will be calculated. Till $a^i = 1$, then i is called the order of a.

   - Example: Find the order of all the nonzero elements in $\mathbb{F}_7$. Solution: $ord(1) = 1$, $ord(2) = 3$, $ord(3) = 6$, $ord(4) = 3$, $ord(5) = 6$, $ord(6) = 2$.

3. The maximal order of an element in $\mathbb{F}_q$ is $q - 1$, and there always exists an element in $\mathbb{F}_q$ such that its order is $q - 1$.

2.4 Elliptic Curve over a Finite Field

2.4.1 Definition

1. General Weierstrass equation for elliptic curves [1]:

$$E: \quad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

   where $a_1, ..., a_6 \in \mathbb{F}$, this equation is usually used in $\mathbb{F}_{2^k}$ or $\mathbb{F}_{3^k}$.

2. An elliptic curve E over a finite field is defined by

$$E: \quad y^2 = x^3 + ax + b \qquad \text{where } a, b \in \mathbb{F} \tag{2}$$

   - Basically, Equation (2) is called the short Weierstrass equation for elliptic curves, in $\mathbb{F}_q$, $\mathbb{F}_{q^k}$ and $q \neq 2$ or 3.
   - The thesis will always work on large prime fields, where the short Weierstrass equation can cover all possible elliptic curves; thus, it will always be used.
   - The thesis only concentrates on an elliptic curve over finite fields. A finite field has only finite many elements and a "curve defined over it should have only finite many points.

3. What does an elliptic curve look like?

   Usually, if elliptic curves are defined over finite field, they look like discrete points sets; hence, the graphs with elliptic curves are demonstrated over $\mathbb{R}$ so that they look like more smoothly.

   Fig. 2.1, Fig. 2.2, Fig. 2.3 and Fig. 2.4 illustrate several different elliptic curves:

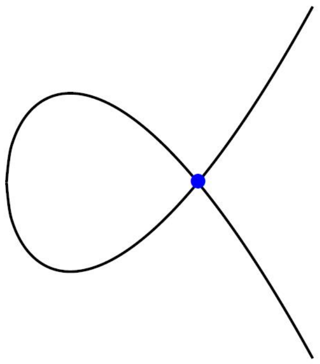


Fig. 2.1: $y^2 = x^3 - 3x + 2$ over $\mathbb{R}$. [1]

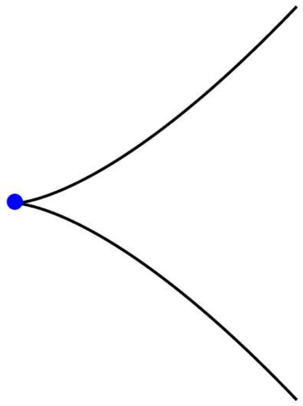


Fig. 2.2: $y^2 = x^3$ over $\mathbb{R}$. [1]

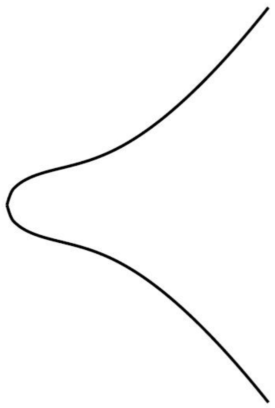


Fig. 2.3: $y^2 = x^3 + x + 1$ over $\mathbb{R}$. [1]

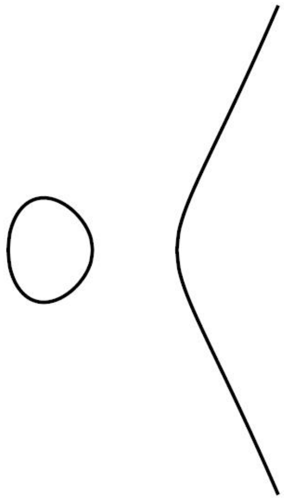


Fig. 2.4: $y^2 = x^3 - x$ over $\mathbb{R}$. [1]

2.4.2 Elliptic Curve Points Operation

A point $P(x_0, y_0)$ on elliptic curve E means: its coordinates x_0 and y_0 are elements in the field, and the coordinates x_0 and y_0 satisfy Equation (2) [26].

1. Elliptic curve points addition:

   Let P, Q and R be three points on an elliptic curve. Points addition $P + Q = R$ can be defined in Fig. 2.5

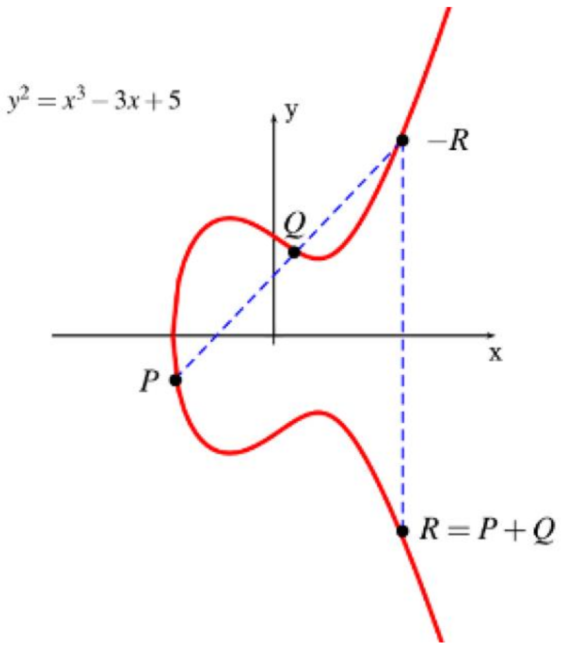


Fig. 2.5: Elliptic Curve Points Addition.

Description: connect P and Q, then extend straight line $\ell_{P,Q}$, it will intersect elliptic curve on another point which is called point $-R$, and then mirror point $-R$ based on *x-axis*, point $R = P + Q$ is obtained.

2. Elliptic curve points doubling:

   Let P, Q be two points on an elliptic curve. Points doubling $P + P = 2P = Q$ can be defined in Fig. 2.6

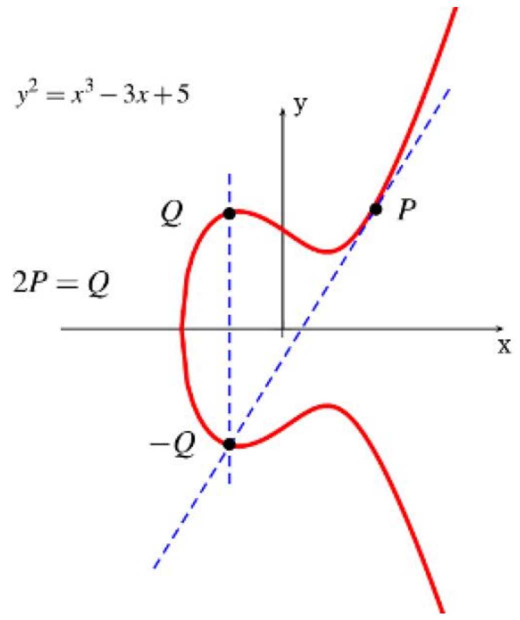


Fig. 2.6: Elliptic Curve Points Doubling.

Description: point P is the tangent point of straight line $\ell_{P,P}$ and elliptic curve, then extend $\ell_{P,P}$, it will intersect elliptic curve on another point which is called point $-Q$, and then mirror point $-Q$ based on x-$axis$, point $Q = 2P$ is obtained.

2.4.3 Elliptic Curve Points Arithmetic

1. Let $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ be two points on the curve $E: \quad y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}$ [27].

   - Assume $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2) \neq \mathcal{O}$, then

     $$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

     where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$, if $P_1 \neq P_2$; and $\lambda = \frac{3x_1^2 + a}{2y_1}$, if $P_1 = P_2$.
   - For any elliptic curve, there exists the point at infinity $\mathcal{O}$, defined by $P + \mathcal{O} = P$, or $P + (-P) = P - P = \mathcal{O}$, for any point $P \in E$.
     - If $x_1 = x_2$ and $y_1 \neq y_2$, then $P_1 = P_2$.

  - If $y_1 = 0$, then $2P_1 = \mathcal{O}$.
  - If $P_1 = (x_1, y_1) \in E$, then $-P_1 = (x_1, -y_1) \in E$.

2. Scalar multiplication [28]:

   - Let P be a point on curve E defined in equation (2)
   - Scalar multiplication nP is defined as

     $nP = \underbrace{P + P + P + ... + P}_{n \text{ times}}$

     where n is an integer; nP *is also a point on the same curve* E.

   - The minimal positive integer a for $aP = \mathcal{O}$ is called the order of P.
   - Scalar multiplication is extensively required in elliptic curve cryptosystems.

2.4.4 The Structure for Points on Elliptic Curve

Let $E(\mathbb{F}_q)$ denote the set of points on elliptic curve E defined over $\mathbb{F}_q$.

1. Number of points on a curve E (the Hasse bound) [29]:

$$\#E(\mathbb{F}_q) = q + 1 - t, \quad |t| \leqslant 2\sqrt{q}. \tag{3}$$

   Note that t is called the trace of Frobenius. It has been shown when q is prime, then every value $N \in [q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$ can be found as a group order $\#E(\mathbb{F}_q)$ for some E.

2. $E(\mathbb{F}_q)$ can be extended to $E(\mathbb{F}_{q^k})$: k is the embedding degree is actually a function $k(q, r)$ of q and r, r is usually the largest order of $E(\mathbb{F}_q)$; thus, k is the smallest positive integer such that $r \mid (q^k - 1)$ [30].

3. The Frobenius endomorphism π is defined as a map from E to E [31]

$$\pi: \ (x, y) \mapsto (x^q, y^q) \tag{4}$$

The Frobenius endomorphism maps any point in $E(\overline{\mathbb{F}}_q)$ to a point in $E(\overline{\mathbb{F}}_q)$, but the set of points fixed by π is the group $E(\mathbb{F}_q)$. As a result, π only does non-trivially on points in $E(\overline{\mathbb{F}}_q)\backslash E(\mathbb{F}_q)$, and more general representation is written as,

$$\pi^i : (x, y) \mapsto (x^{q^i}, y^{q^i}) \tag{5}$$

acts non-trivially on points in $E(\overline{\mathbb{F}}_q)\backslash E(\mathbb{F}_{q^i})$.

Note that, $E(\overline{\mathbb{F}}_q)$ is a large set, which can be called $E(\mathbb{F}_{q^k})$ where k is the embedding degree. Namely, $E(\mathbb{F}_q) \subset E(\mathbb{F}_{q^2}) \subset E(\mathbb{F}_{q^3}) \subset$, ..., $\subset E(\mathbb{F}_{q^{k-1}}) \subset E(\mathbb{F}_{q^k})$:

- $E(\overline{\mathbb{F}}_q)\backslash E(\mathbb{F}_q)$ means the set $E(\overline{\mathbb{F}}_q)$ only excludes $E(\mathbb{F}_q)$.
- Likewise, $E(\overline{\mathbb{F}}_q)\backslash E(\mathbb{F}_{q^i})$ means the set $E(\overline{\mathbb{F}}_q)$ just excludes $E(\mathbb{F}_{q^i})$, where $1 < i < k$.

2.4.5 Basics on Analytic Geometry

Let elliptic curve E be given as

$$E: \quad y^2 = x^3 + ax + b$$

And also let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on E.

1. Let the chord line $\ell_{P,Q}$ joining P and Q be $y = kx + d$. Then k and d can be

solved as

$$k = \frac{y_2 - y_1}{x_2 - x_1} \quad \text{and}$$

$$d = y_1 - kx_1 = \frac{x_2y_1 - x_1y_2}{x_2 - x_1}$$

So it follows

$$\ell_{P,Q}: \quad y = \frac{y_2 - y_1}{x_2 - x_1} \cdot x + \frac{x_2y_1 - x_1y_2}{x_2 - x_1} \tag{6}$$

Since in Chapter 4, Chapter 5 and Chapter 6, the Miller's Algorithm with the straight lines will be calculated, which includes the parameters of coordinates x and y; consequently, Equation (6) can be written as

$$\ell_{P,Q}: \quad y - \frac{y_2 - y_1}{x_2 - x_1} \cdot x - \frac{x_2y_1 - x_1y_2}{x_2 - x_1} \tag{7}$$

2. Let the tangent line $\ell_{P,P}$ to E at point P be given as $y = k'x + d'$. Then k' can be solved as follows:

   First find derivative of E at point P:

$$\begin{aligned} (y^2)'_x &= (x^3 + ax + b)'_x \\ 2yy'_x &= 3x^2 + a \end{aligned}$$

   It follows

$$k' = y'_x|_P = \left.\frac{3x^2 + a}{2y}\right|_{(x,y)=(x_1,y_1)} = \frac{3x_1^2 + a}{2y_1}$$

   then

$$d' = y_1 - k'x_1 = y_1 - \frac{3x_1^2 + a}{2y_1} \cdot x_1 = \frac{2y_1^2 - 3x_1^3 - ax_1}{2y_1} = \frac{-y_1^2 + 2ax_1 + 3b}{2y_1}$$

   Hence,

$$\ell_{P,P}: \quad y = \frac{3x_1^2 + a}{2y_1} \cdot x + \frac{-y_1^2 + 2ax_1 + 3b}{2y_1} \tag{8}$$

Similarly, Equation (8) has the other representation,

$$\ell_{P,P}: \quad y - \frac{3x_1^2 + a}{2y_1} \cdot x - \frac{-y_1^2 + 2ax_1 + 3b}{2y_1} \tag{9}$$

3. The vertical line ν_Q at point Q can be given as

$$\nu_Q: \; x = x_2 \tag{10}$$

For the same reason, Equation (10) can be represented as

$$\nu_Q: \; x - x_2 \tag{11}$$

The straight lines $\ell_{P,Q}$, $\ell_{P,P}$ and ν_Q are represented by Equation (7), (9) and (11), respectively; so that they are conveniently substituted in the algorithms of the following chapters.

3 DIVISORS AND BILINEAR MAP

3.1 Divisors

Basically, divisors have wide definitions in algebraic geometry field, but this thesis just concentrates on the parts which are used in the understanding of cryptographic pairing computations [32].

3.1.1 Definition

A divisor D on curve E is a convenient way to denote a multi-set of points on E, written as the formal sum [1]

$$D = \sum_{P \in E(\bar{F}_q)} n_P(P), \quad \text{where } n_P \in \mathbb{Z}.$$

- The set of all divisors on E is denoted by $\mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$ and forms a group, where addition of divisors is natural.
- The zero divisor: it is the divisor with all $n_P = 0$, the zero divisor $0 \in \mathrm{Div}_{\overline{\mathbb{F}}_q}(E)$.
- If the field $\overline{\mathbb{F}}_q$ is not specific, it can be omitted and simply written as $\mathrm{Div}(E)$ to denote the group of divisors.

A divisor D on curve E denotes the multiplicities of points on E; in other words, it can represent a kind of relationship of lines and elliptic curve; moreover, it is the cornerstone of pairing-based algorithms.

3.1.2 The Degree and Support of D

1. The degree of a divisor D is $Deg(D) = \sum_{P \in E(\overline{\mathbb{F}}_q)} n_P$,

2. The support of D, denoted by the set

$$\text{supp}(D) = \{P \in E(\overline{\mathbb{F}}_q) : n_P \neq 0\}.$$

For instance,

Let $P, Q, R, S \in E(\overline{\mathbb{F}}_q)$. Let $D_1 = 3(P) - 4(Q)$, and $D_2 = 4(Q) + (R) - 2(S)$, so the $Deg(D_1) = 3 - 4 = -1$, and $Deg(D_2) = 4 + 1 - 2 = 3$. The sum $D_1 + D_2 = 3(P) + (R) - 2(S)$, and naturally $Deg(D_1 + D_2) = Deg(D_1) + Deg(D_2) = 2$.

The supports are $supp(D_1) = \{P, Q\}$, $supp(D_2) = \{Q, R, S\}$, and $supp(D_1{+}D_2) = \{P, R, S\}$.

3.1.3 The Divisor of a Function f on E

1. The divisor of a function f on E is used to denote the intersection points (and their multiplicities) of f and E.

   - Let $\text{ord}_P(f)$ count the multiplicity of f at P, which is positive if f has a zero at P, and negative if f has a pole at P. The divisor of a function f is defined as
   $$(f) = \sum_{P \in E(\overline{\mathbb{F}}_q)} \text{ord}_P(f)(P).$$
   - Notice that in all cases, $\text{Deg}((\ell)) = 0$. In fact, this is true for any function f on E.

2. The relationship of a function f and a divisor D:

   A divisor $D = \sum_P n_P(P)$ is a divisor of a function if and only if
   $$\sum_P n_P = 0 \text{ and } \sum_P [n_P]P = \mathcal{O} \text{ on } E.$$

   For example,

Let f be a line that intersects E at P and Q. Then divisor $(f) = (\ell_{P,Q}) = (P) + (Q) + ([-1](P+Q)) - 3(\mathcal{O})$, since

$$\sum_P n_P = n_P + n_Q + n_{[-1](P+Q)} + n_{\mathcal{O}} = 1 + 1 + 1 - 3 = 0$$

$$\sum_P [n_P]P = P + Q + ([-1](P+Q)) = \mathcal{O} \text{ (Elliptic Curve Points Operation)}$$

3. There are three scenarios that straight line f intersects curve E.

   (a) In Fig. 3.1, the chord line $\ell_{P,Q}$ intersects E in P, Q and $[-1](P+Q)$, all with multiplicity 1, and $\ell_{P,Q}$ also intersects E with multiplicity -3 at $\mathcal{O}$, namely, $\ell_{P,Q}$ has a pole of order 3 at $\mathcal{O}$. Thus, $\ell_{P,Q}$ has divisor

$$(\ell_{P,Q}) = (P) + (Q) + ([-1](P+Q)) - 3(\mathcal{O}). \tag{12}$$

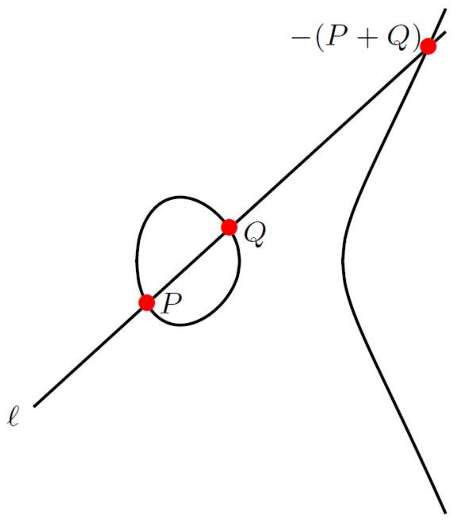


Fig. 3.1: The function $(\ell_{P,Q})$

   (b) In Fig. 3.2, the tangent line $\ell_{P,P}$ intersects E with multiplicity 2 at P, with multiplicity 1 at $[-2]P$, and again with multiplicity -3 at $\mathcal{O}$, so in this case

$$(\ell_{P,P}) = 2(P) + ([-2]P) - 3(\mathcal{O}). \tag{13}$$

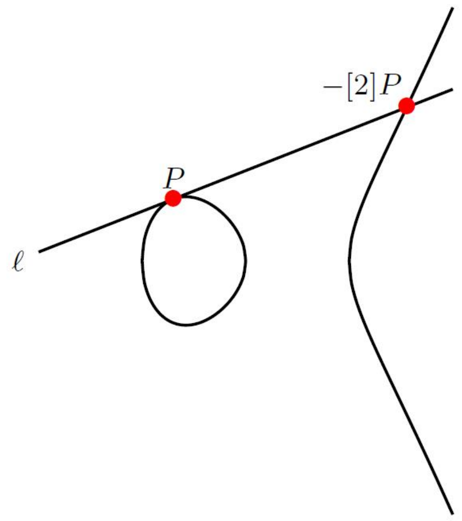


Fig. 3.2: The function $(\ell_{P,P})$

(c) In Fig. 3.3, the vertical line ν_{P+Q} intersects E in $(P+Q)$ and $[-1](P+Q)$ with multiplicity 1.

$$(\nu_{P+Q}) = ((P+Q)) + ([-1](P+Q)) - 2(\mathcal{O}). \tag{14}$$

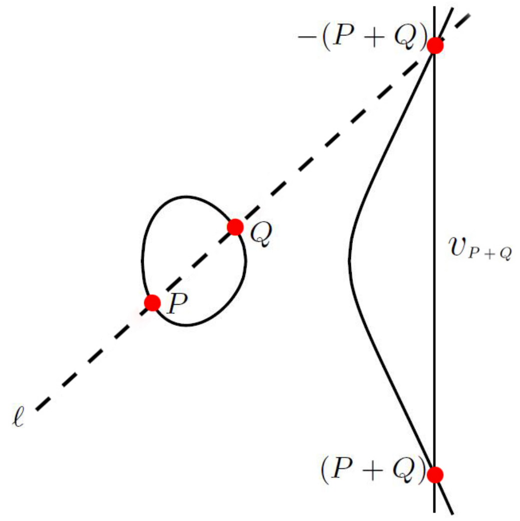


Fig. 3.3: The function (ν_{P+Q})

4. Properties of divisors of the functions:

   (a) $(fg) = (f) + (g)$

   (b) $(f/g) = (f) - (g)$

   (c) $(f) = 0$ if and only if f is constant.

   (d) If $(f) = (g)$, then $(f/g) = 0$, so f is a constant multiple of g.

3.1.4 Equivalence of Divisors

The divisors D_1 and D_2 can be called equivalent, written as $D_1 \sim D_2$, $D_1 = D_2 + (f)$ for some function f. The notion of equivalence allows us to reduce divisors of any size D into much smaller divisors.

For instance,

- Let $R = P+Q$ on E, so the line ℓ joining P and Q have divisor $(\ell) = (P)+(Q)+(-R)-3(\mathcal{O})$, whilst the vertical line $\nu = x - x_R$ has divisor $(\nu) = (-R)+(R)-2(\mathcal{O})$. In addition, the quotient ℓ/ν has divisor $(\frac{\ell}{\nu}) = (P) + (Q) - (R) - (\mathcal{O})$.

  Thus, the equation $R = P+Q$ on E is the same as the divisor equality $(R)-(\mathcal{O}) = (P) - (\mathcal{O}) + (Q) - (\mathcal{O}) - (\frac{\ell}{\nu})$. It reduces $(P) + (Q) - 2(\mathcal{O})$ to $(R) - (\mathcal{O})$

- Similarly, in order to obtain $([2]Q)-(Q) = (Q)-(\mathcal{O})$, there exists a $(f) = 2(Q)-([2]Q) - (\mathcal{O})$. This equivalence will be used in following chapters' algorithms to substitute $D_Q = (Q) - (\mathcal{O})$ with $D_Q = ([2]Q) - (Q)$, so that it is convenient to compute D_Q using $[2]Q$ and Q, rather than Q and $\mathcal{O}$.

3.2 Bilinear Map

3.2.1 Definition

In pairing-based cryptography, bilinear map [33] plays a central role, it maps elements of two cryptographic groups to a third group, in many literatures, it is written as

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$$

Usually, bilinear map defines the groups $\mathbb{G}_1$ in $E(\mathbb{F}_q)$, $\mathbb{G}_2$ in $E(\mathbb{F}_{q^k})/E(\mathbb{F}_q)$, as well as the target group $\mathbb{G}_T$ in the multiplicative group $\mathbb{F}^*_{q^k}$, so it can be called $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive, whilst $\mathbb{G}_T$ is multiplicative [34].

If points P and Q are the elements of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Then bilinear map can be rewritten as

$$e(P, Q) : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$$

where $P \in \mathbb{G}_1 = E(\mathbb{F}_q)$, $Q \in \mathbb{G}_2 = E(\mathbb{F}_{q^k})/E(\mathbb{F}_q)$, and $e(P, Q) \in \mathbb{G}_T = \mathbb{F}^*_{q^k}$.

3.2.2 Properties

In many literatures, the properties of bilinear map are mentioned as,

- For $P,\ P' \in \mathbb{G}_1$ and $Q,\ Q' \in \mathbb{G}_2$,

$$e(P + P', Q) = e(P, Q) \cdot e(P', Q),$$

$$e(P, Q + Q') = e(P, Q) \cdot e(P, Q').$$

- For scalars $a, b \in \mathbb{Z}$,

$$e(P, 0) = e(0, Q) = 1,$$

$$e(-P, Q) = e(P, Q)^{-1} = e(P, -Q),$$

$$e([a]P, Q) = e(P, Q)^a = e(P, [a]Q),$$

$$e([a]P, [b]Q) = e(P, [b]Q)^a = e([a]P, Q)^b = e(P, Q)^{ab} = e([b]P, [a]Q).$$

3.2.3 Solve the Decision Diffie-Hellman (DDH) Problem with the Properties

Bilinear map was initially found as an useful tool in cryptanalysis [35]; for instance, it can solve the Decision Diffie-Hellman (DDH) Problem [36] [37]. In other words, bilinear map can reduce the discrete logarithm problem on elliptic curves or hyperelliptic curves [18].

The Decision Diffie-Hellman (DDH) problem is: Given $P, [a]P, b[P]$, and Q, determine whether $Q = [ab]P$ or not [37].

Pairings make the DDH problem "easy":

1) Compute $e(P, Q) = A$,

2) Compute $e([a]P, [b]P) = B$,

3) $Q = [ab]P$ if and only if $A = B$.

Because $e([a]P, [b]P) = e(P, P)^{ab} = e(P, [ab]P)$, if $e(P, [ab]P) = e(P, Q)$, then $Q = [ab]P$.

3.2.4 Implementation Methods of $e(P, Q)$

There are two popular methods to implement $e(P, Q)$:

- Weil pairing $w_r(P, Q)$
- Tate pairing $t_r(P, Q)$ [38]

In this thesis, it concentrates on Weil pairing $w_r(P, Q)$ to implement bilinear map. Additionally, Miller's algorithm is the core algorithm to compute $w_r(P, Q)$.

Consequently, in next chapter, it will move to Weil pairing and Miller's algorithm.

4 AN OVERVIEW OF MILLER'S ALGORITHM AND RELATED WORKS

4.1 Weil Pairing

The Weil pairing (over finite fields):

Let $P, Q \in E(\mathbb{F}_{q^k})[r]$ and let D_P and D_Q be *degree zero divisors* with *disjoint supports* such that $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$. There exist functions f and g such that $(f) = rD_P$ and $(g) = rD_Q$.

w_r is a map:

$$w_r : E(\mathbb{F}_{q^k})[r] \times E(\mathbb{F}_{q^k})[r] \mapsto \mathbb{F}_{q^k}[r],$$

defined as:

$$w_r(P, Q) = \frac{f(D_Q)}{g(D_P)} \tag{15}$$

- Note that r is the largest prime factor of $\#E(\mathbb{F}_q)$.
- For a point $P \in E(\mathbb{F}_{q^k})[r]$, the function $f = f_{r,P}$ with divisor $r(P) - r(\mathcal{O})$ plays the major role in the Weil pairing definition.
- Likewise, for a point $Q \in E(\mathbb{F}_{q^k})[r]$, the function $g = g_{r,Q}$ has the divisor $r(Q) - r(\mathcal{O})$.
- According to Equation (15), $w_r(P, Q)$ equals to $f(D_Q)$ divides $g(D_P)$. Moreover, $f(D_Q)$ and $g(D_P)$ can be calculated with Miller's Algorithm, respectively. Consequently, this chapter just performs how calculates $f(D_Q)$ using Miller's Algorithm; on the other hand, under the Miller's Algorithm, $g(D_P)$ can use the similar method to compute.

4.2 Miller's Algorithm

In this section, it is necessary to recall the knowledge of divisors and the function f. Some important equations will be deducted:

- For any $m \in \mathbb{Z}$ and $P \in E$, it follows that there exists a function $f_{m,P}$ with divisor
$$(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O}), \tag{16}$$
  - where it is noted that for $m = 0$, it can take $f_{0,P} = 1$ with $(f_{0,P})$ the zero divisor.
$$(f_{0,P}) = 0(P) - ([0]P) - (0-1)(\mathcal{O}) = 0,$$
  - where it is noted that for $m = 1$, it can take $f_{1,P} = 1$ with $(f_{1,P})$ the zero divisor.
$$(f_{1,P}) = 1(P) - ([1]P) - (1-1)(\mathcal{O}) = 0,$$
  - Note: $(f_{0,P}) = (f_{1,P}) =$ zero divisor, according to "Properties of divisors of functions: $(f) = 0$ if and only if f is constant", so it is convenient to take $f_{0,P} = f_{1,P} = 1$ for setting the initial value.

- From $f_{m,P}$ to $f_{m+1,P}$:
  - When $P \in E[r]$, means r is the order of P, then following (16), $f_{r,P}$ has divisor
$$(f_{r,P}) = r(P) - r(\mathcal{O}). \tag{17}$$

    Furthermore,

$$(f_{m+1,P}) = (m+1)(P) - ([m+1]P) - (m)(\mathcal{O}), \tag{18}$$

Observe that, Equation (18) subtracts Equation (16), then acquire

$$(f_{m+1,P}) - (f_{m,P}) = (P) + ([m]P) - ([m+1]P) - (\mathcal{O}). \tag{19}$$

– In Fig. 4.1, according to the functions of chord line and vertical line, it is obtained

$$(\ell_{[m]P,P}) = (P) + ([m]P) + (-[m+1]P) - 3(\mathcal{O}),$$

$$(\nu_{[m+1]P}) = (-[m+1]P) + ([m+1]P) - 2(\mathcal{O}),$$

Thus,

$$(\ell_{[m]P,P}/\nu_{[m+1]P}) = (\ell_{[m]P,P}) - (\nu_{[m+1]P}) = (P) + ([m]P) - ([m+1]P) - (\mathcal{O}) \tag{20}$$

where $\ell_{[m]P,P}$ and $\nu_{[m+1]P}$ are the chord and vertical lines used in the chord-and-tangent addition of the point $[m]P$ and P.

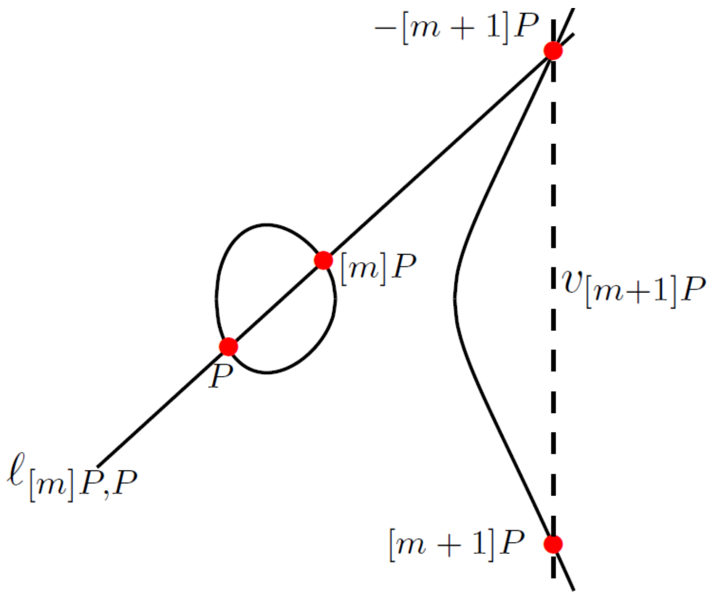


Fig. 4.1: A Function: $(\ell_{[m]P,P}/\nu_{[m+1]P})$

From (19) and (20) it can be seen that $(f_{m+1,P}) - (f_{m,P})$ is exactly the divisor of the function $\ell_{[m]P,P}/\nu_{[m+1]P}$, which means $f_{m+1,P}$ can be built from $f_{m,P}$ via

$$f_{m+1,P} = f_{m,P} \times \frac{\ell_{[m]P,P}}{\nu_{[m+1]P}} \tag{21}$$

- From $f_{m,P}$ to $f_{2m,P}$:
  - In addition, according to Properties " $(fg) = (f) + (g)$ ":

    $$(f_{m,P}^2) = (f_{m,P} \times f_{m,P}) = (f_{m,P}) + (f_{m,P}) = 2(f_{m,P}),$$

    Hence, following (16)

    $$(f_{m,P}^2) = 2(f_{m,P}) = 2m(P) - 2([m]P) - 2(m-1)(\mathcal{O});$$

    Moreover, also following (16)

    $$(f_{2m,P}) = 2m(P) - ([2m]P) - (2m-1)(\mathcal{O}),$$

    Observe that,

    $$(f_{2m,P}) - (f_{m,P}^2) = 2([m]P) - ([2m]P) - (\mathcal{O}).$$

  - Now, the functions of chord line and vertical line can be rewritten as:

    $$(\ell_{[m]P,[m]P}) = ([m]P) + ([m]P) + (-[2m]P) - 3(\mathcal{O}),$$

    $$(\nu_{[2m]P}) = (-[2m]P) + ([2m]P) - 2(\mathcal{O}),$$

Similarly, according to (20), it is obtained

$$(\ell_{[m]P,[m]P}/\nu_{[2m]P}) = (\ell_{[m]P,[m]P}) - (\nu_{[2m]P}) = 2([m]P) - ([2m]P) - (\mathcal{O}),$$

Therefore,

$$(f_{2m,P}) - (f_{m,P}^2) = 2([m]P) - ([2m]P) - (\mathcal{O}) = (\ell_{[m]P,[m]P}/\nu_{[2m]P})$$

At last,

$$f_{2m,P} = f_{m,P}^2 \times \frac{\ell_{[m]P,[m]P}}{\nu_{[2m]P}} \tag{22}$$

– Based on (22), it can be straightly jumped from $f_{m,P}$ to $f_{2m,P}$, in comparison with the naive method of progressing one-by-one in Fig. 4.2:

$$\begin{tikzcd}
f_{m,P} \arrow[r, "\cdot\frac{\ell_{[m]P,P}}{v_{[m+1]P}}"] \arrow[rrrrr, bend right, dashed, "f_{m,P}^2\cdot\frac{\ell_{[m]P,[m]P}}{v_{[2m]P}}"'] & f_{m+1,P} \arrow[r, "\cdot\frac{\ell_{[m+1]P,P}}{v_{[m+2]P}}"] & \cdots & \cdots \arrow[r, "\cdot\frac{\ell_{[2m-2]P,P}}{v_{[2m-1]P}}"] & f_{2m-1,P} \arrow[r, "\cdot\frac{\ell_{[2m-1]P,P}}{v_{[2m]P}}"] & f_{2m,P}
\end{tikzcd}$$

Fig. 4.2: Jump from $f_{m,P}$ to $f_{2m,P}$ [1]

So far, for any m, either $f_{m+1,P}$, or $f_{2m,P}$ can be obtained quickly, Miller observed that, then gives rise to a double-and-add style algorithm.

1. This is the Miller's Algorithm, then an example will be given to demonstrate how it computes.

Algorithm 4.1 Miller's Algorithm [2]

Input: $P \in E(\mathbb{F}_{q^k})[r], D_Q \sim (Q) - (\mathcal{O})$ with support disjoint from $(f_{r,P})$, and $r = (r_{n-1}...r_1r_0)_2$ with $r_{n-1} = 1$.

Output: $f_{r,P}(D_Q) \leftarrow f$.

```
1: $R \leftarrow P, f \leftarrow 1$.
2: for $i = n - 2$ down to 0 do
3:   Compute the line function $\ell_{R,R}$.
4:   $R \leftarrow [2]R$.
5:   Compute the line function $\nu_R$.
6:   $f \leftarrow f^2 \times \frac{\ell_{R,R}}{\nu_R}(D_Q)$.
7:   if $r_i = 1$, then
8:     Compute the line function $\ell_{R,P}$.
9:     $R \leftarrow R + P$.
10:    Compute the line function $\nu_R$.
11:    $f \leftarrow f \times \frac{\ell_{R,P}}{\nu_R}(D_Q)$.
12:  end if
13: end for
14: return $f$.
```

- Steps 3-6 of Algorithm 4.1 can be called a **doubling stage**, which is different from elliptic curve points' doubling operation.
- Steps 7-12 of Algorithm 4.1 can be called an **addition stage**, which is different from elliptic curve points' addition operation.
- The algorithm calculates r from the most significant digit to the least significant digit (where r is a binary number of length n), namely, the sequence of computation is from left to right.

2. The details of the computation:

   Input: $P,\ D_Q,\ r = 29 = (11101)_2$

   Output: $f_{29,P}(D_Q) \leftarrow f$

   Compute:

   (a) $P;\quad f_{1,P} = 1,$

   (b) $r_3 = 1$:

i. $\ell_{P,P}$

ii. $2P = 2 \times P$

iii. ν_{2P}

iv. $f_{2,P} = f_{1,P}^2 \times \frac{\ell_{P,P}}{\nu_{2P}}(D_Q)$

v. $\ell_{2P,P}$

vi. $3P = 2P + P$

vii. ν_{3P}

viii. $f_{3,P} = f_{2,P} \times \frac{\ell_{2P,P}}{\nu_{3P}}(D_Q)$

(c) $r_2 = 1$:

i. $\ell_{3P,3P}$

ii. $6P = 2 \times 3P$

iii. ν_{6P}

iv. $f_{6,P} = f_{3,P}^2 \times \frac{\ell_{3P,3P}}{\nu_{6P}}(D_Q)$

v. $\ell_{6P,P}$

vi. $7P = 6P + P$

vii. ν_{7P}

viii. $f_{7,P} = f_{6,P} \times \frac{\ell_{6P,P}}{\nu_{7P}}(D_Q)$

(d) $r_1 = 0$:

i. $\ell_{7P,7P}$

ii. $14P = 2 \times 7P$

iii. ν_{14P}

iv. $f_{14,P} = f_{7,P}^2 \times \frac{\ell_{7P,7P}}{\nu_{14P}}(D_Q)$

(e) $r_0 = 1$:

i. $\ell_{14P,14P}$

ii. $28P = 2 \times 14P$

iii. ν_{28P}

iv. $f_{28,P} = f_{14,P}^2 \times \frac{\ell_{14P,14P}}{\nu_{28P}}(D_Q)$

v. $\ell_{28P,P}$

vi. $29P = 28P + P$

vii. ν_{29P}

viii. $f_{29,P} = f_{28,P} \times \frac{\ell_{28P,P}}{\nu_{29P}}(D_Q)$

(f) return $f_{29,P}$

- Note:
  - The steps v to viii of (e) need to be noticed, because $r = 29$ is the order of point P, that means $29P = \mathcal{O}$. As a result, $\mathcal{O} = 29P = 28P + P$, namely, $28P = -P$.
  - As a consequence, in step v of (e), it can be written $\ell_{28P,P} = \ell_{-P,P}$. Based on the geometry, the chord line $\ell_{-P,P}$ is just the vertical line ν_P. Fortunately, it just conveniently computes ν_P instead of $\ell_{28P,P}$ in step v of (e).
  - In step vi of (e), $\mathcal{O} = 29P = 28P + P$; thus, this step doesn't need to be computed.
  - In projective plane, $\mathcal{O}$ is defined as $\mathcal{O} = (0 : 1 : 0)$, and

$$\nu_{\mathcal{O}} : \quad y = 1,$$

    in step vii of (e), $\nu_{29P} = \nu_{\mathcal{O}}$, so ν_{29P} equals to constant 1.
  - Finally, in step viii of (e),

$$f_{29,P} = f_{28,P} \times \frac{\ell_{28P,P}}{\nu_{29P}}(D_Q) = f_{28,P} \times \frac{\nu_P}{1}(D_Q) = f_{28,P} \times \nu_P(D_Q)$$

In generalization, when computes the last loop of Miller's Algorithm, it is able to use

$$\nu_P \text{ instead of } \ell_{[r-1]P,P}$$

and

$$\text{constant 1 instead of } \nu_{[r]P}$$

to simplify the last several steps.

Hence, the last step can be written as:

$$f_{r,P} = f_{r-1,P} \times \frac{\ell_{[r-1]P,P}}{\nu_{[r]P}}(D_Q) = f_{r-1,P} \times \frac{\nu_P}{1}(D_Q) = f_{r-1,P} \times \nu_P(D_Q)$$

Additionally, $\mathcal{O} = rP$ doesn't have to be computed.

3. Now, it is time to explain how to compute $\frac{\ell}{\nu}(D_Q)$:

   (a) $D_Q \sim (Q) - (\mathcal{O})$, based on the concept of "Equivalence of Divisors" in the Subsection (3.1.4) at the Page 24, $([2]Q) - (Q) \sim (Q) - (\mathcal{O})$, so equation $D_Q = ([2]Q) - (Q)$ can be obtained. Although using this equivalence "$D_Q = ([2]Q) - (Q)$", it still follows the input restriction of Miller's Algorithm (P and D_Q with support disjoint from $(f_{r,P})$).

   (b) Functions ℓ and ν can be calculated respectively, according to the Subsection (2.4.5) at the Page 17.

   (c) Under the divisor theory:

$$\frac{\ell}{\nu}(D_Q) = \frac{\ell(D_Q)}{\nu(D_Q)} = \frac{\ell(([2]Q) - (Q))}{\nu(([2]Q) - (Q))}$$

   moreover,

$$\ell(([2]Q) - (Q)) = \frac{\ell([2]Q)}{\ell(Q)}; \quad \nu(([2]Q) - (Q)) = \frac{\nu([2]Q)}{\nu(Q)}$$

Thus,

$$\frac{\ell}{\nu}(D_Q) = \frac{\ell([2]Q)}{\ell(Q)} \times \frac{\nu(Q)}{\nu([2]Q)} = \frac{\ell([2]Q) \times \nu(Q)}{\ell(Q) \times \nu([2]Q)} \tag{23}$$

(d) At last, correspondingly substituting the x-coordinates and y-coordinates of point $[2]Q$ and point Q into the functions ℓ and ν in Equation (23), the result of $\frac{\ell}{\nu}(D_Q)$ can be gotten.

Therefore, these steps are the details of calculating $\frac{\ell}{\nu}(D_Q)$.

4. There is another existing work, Miller's Algorithm Using Signed Digit Number [3]. However, its one problem makes it cannot work, and the problem will be corrected in next chapter.

5 PROPOSED WORKS

In this chapter, the correction of existing work "Miller's Algorithm Using Signed Digit Number" will be attested. Moreover, two new algorithms will be proposed: the New Right-to-left Miller's Algorithm and the Modified Miller's Algorithm with Enhanced Security. These two algorithms have different features and usages. In addition, their examples and contributions will also be demonstrated.

5.1 The Correction of Miller's Algorithm Using Signed Digit Number

In every loop of Miller's Algorithm, if $r_i = 0$, then it just needs a doubling stage (steps 3-6 of Algorithm 4.1); whilst, if $r_i = 1$, then it needs a doubling stage plus an addition stage (steps 7-12 of Algorithm 4.1), that means the total steps are steps 3-12 of Algorithm 4.1. Consequently, when $r_i = 1$, the Algorithm needs around twice computation steps in comparison with $r_i = 0$ [39].

Hence, when the length of $r = (r_{n-1}...r_1r_0)_2$ is not changed, increasing the the number of zero and decreasing the the number of nonzero, it is able to keep the number of the doubling stages unchanged and lessen the number of the addition stages [40]; as a result, calculation will be more efficient [41].

For this motivation, *Miller's Algorithm Using Signed Digit Number* substitutes binary number system with signed digit number system, and the signed digit number system can increase the number of zero and reduce the the number of nonzero, so it is able to decrease the calculation steps relatively, such that it can make the computation more efficient [42].

Nevertheless, the Algorithm 5.1 is wrong when $r_i = \overline{1}$, it can be attested.

Algorithm 5.1 Miller's Algorithm Using Signed Digit Number [3]

Input: $P \in E(\mathbb{F}_{q^k})[r], D_Q \sim (Q) - (\mathcal{O})$ with support disjoint from $(f_{r,P})$, and $r = (r_{n-1}...r_1r_0)_2$ with $r_{n-1} = 1$. Additionally, $r_i \in \{\overline{1}, 0, 1\}$.

Output: $f_{r,P}(D_Q) \leftarrow f$.

```
1: $R \leftarrow P, f \leftarrow 1$.
2: for $i = n - 2$ down to 0 do
3:     Compute the line function $\ell_{R,R}$.
4:     $R \leftarrow [2]R$.
5:     Compute the line function $\nu_R$.
6:     $f \leftarrow f^2 \times \frac{\ell_{R,R}}{\nu_R}(D_Q)$.
7:     if $r_i = 1$, then
8:         Compute the line function $\ell_{R,P}$.
9:         $R \leftarrow R + P$.
10:        Compute the line function $\nu_R$.
11:        $f \leftarrow f \times \frac{\ell_{R,P}}{\nu_R}(D_Q)$.
12:    end if
13:    if $r_i = \overline{1}$, then
14:        Compute the line function $\nu_R$.
15:        $R \leftarrow R - P$.
16:        Compute the line function $\ell_{-R,P}$  *
17:        $f \leftarrow f \times \frac{\nu_R}{\ell_{-R,P}}(D_Q)$.  *
18:    end if
19: end for
20: return $f$.
```

* *These two steps of this algorithm [3] are wrong when $r_i = \overline{1}$, it will be corrected when calculating the step of $r_i = \overline{1}$.*

Likewise, the algorithm calculates r from the most significant digit to the least significant digit (where r is a binary number of length n), namely, the sequence of computation is also from left to right.

1. *Correct the Miller's Algorithm Using Signed Digit Number*

   When $r_i = \overline{1}$, namely, it needs to be calculated the function from $f_{m+1,P}$ to $f_{m,P}$:

   Based on (21),

$$f_{m+1,P} = f_{m,P} \times \frac{\ell_{[m]P,P}}{\nu_{[m+1]P}}$$

Acquire that,

$$f_{m,P} = f_{m+1,P} \times \frac{\nu_{[m+1]P}}{\ell_{[m]P,P}} \tag{24}$$

Therefore, when $r_i = \overline{1}$, the corrected steps 16 and 17 should be

```
16:     Compute the line function $\ell_{R,P}$.
17:     $f \leftarrow f \times \frac{\nu_R}{\ell_{R,P}}(D_Q)$.
```

2. *It will be exemplified the correction is accurate, but the two significant steps of the original Miller's Algorithm Using Signed Digit Number are wrong.*

- This computation follows the correction:

  Input: $P,\ D_Q,\ r = 29 = (100\overline{1}01)_2$

  Output: $f_{29,P}(D_Q) \leftarrow f$

  Compute:

  (a) $P;\quad f_{1,P} = 1,$

  (b) $r_4 = 0$:

  i. $\ell_{P,P}$

  ii. $2P = 2 \times P$

  iii. ν_{2P}

  iv. $f_{2,P} = f_{1,P}^2 \times \frac{\ell_{P,P}}{\nu_{2P}}(D_Q)$

  (c) $r_3 = 0$:

  i. $\ell_{2P,2P}$

  ii. $4P = 2 \times 2P$

  iii. ν_{4P}

  iv. $f_{4,P} = f_{2,P}^2 \times \frac{\ell_{2P,2P}}{\nu_{4P}}(D_Q)$

(d) $r_2 = \overline{1}$: *these steps of (d) are different from the original Miller's Algorithm.*

   i. $\ell_{4P,4P}$

   ii. $8P = 2 \times 4P$

   iii. ν_{8P}

   iv. $f_{8,P} = f_{4,P}^2 \times \frac{\ell_{4P,4P}}{\nu_{8P}}(D_Q)$

   v. ν_{8P}

   vi. $7P = 8P - P$

   vii. $\ell_{7P,P}$

   viii. $f_{7,P} = f_{8,P} \times \frac{\nu_{8P}}{\ell_{7P,P}}(D_Q)$

(e) $r_1 = 0$:

   i. $\ell_{7P,7P}$

   ii. $14P = 2 \times 7P$

   iii. ν_{14P}

   iv. $f_{14,P} = f_{7,P}^2 \times \frac{\ell_{7P,7P}}{\nu_{14P}}(D_Q)$

(f) $r_0 = 1$:

   i. $\ell_{14P,14P}$

   ii. $28P = 2 \times 14P$

   iii. ν_{28P}

   iv. $f_{28,P} = f_{14,P}^2 \times \frac{\ell_{14P,14P}}{\nu_{28P}}(D_Q)$

   v. $\ell_{28P,P}$

   vi. $29P = 28P + P$

   vii. ν_{29P}

   viii. $f_{29,P} = f_{28,P} \times \frac{\ell_{28P,P}}{\nu_{29P}}(D_Q)$

(g) return $f_{29,P}$

The last steps v to viii of (f) are also able to follow the Note in the Subsection (4.2) at the Page 34 to calculate.

Thus, following the correction can smoothly obtain the final result.

- If the computations follow the *Miller's Algorithm Using Signed Digit Number*, then the steps in (d) of the correction will be:

  (d) $r_2 = \overline{1}$:

  i. $\ell_{4P,4P}$

  ii. $8P = 2 \times 4P$

  iii. ν_{8P}

  iv. $f_{8,P} = f_{4,P}^2 \times \frac{\ell_{4P,4P}}{\nu_{8P}}(D_Q)$

  v. ν_{8P}

  vi. $7P = 8P - P$

  vii. $\ell_{-7P,P}$ *

  viii. $f_{7,P} = f_{8,P} \times \frac{\nu_{8P}}{\ell_{-7P,P}}(D_Q)$ *

  There are two important aspects that incur the inaccuracy of the *Miller's Algorithm Using Signed Digit Number*:

  – $r = 29$ is the order of point P, that means $\mathcal{O} = 29P = 22P + 7P$, namely, $22P = -7P$. In this way, $\ell_{-7P,P}$ could be $\ell_{22P,P}$ in step vii. On the other hand, when it calculates the point $7P$, it cannot jump to reckon the point $22P$ because both Elliptic Curve points operation and Miller's Algorithm are accumulative computations. What is more, if calculating $\ell_{-7P,P}$ in step vii, then the step vi ($7P = 8P - P$) is useless. Therefore, in this circumstance, reckoning $\ell_{-7P,P}$ is neither accurate nor practical.

– According to Equation (24)

$$f_{m,P} = f_{m+1,P} \times \frac{\nu_{[m+1]P}}{\ell_{[m]P,P}}$$

$f_{7,P}$ should equal to

$$f_{8,P} \times \frac{\nu_{8P}}{\ell_{7P,P}}(D_Q)$$

neither

$$f_{8,P} \times \frac{\nu_{8P}}{\ell_{-7P,P}}(D_Q)$$

nor

$$f_{8,P} \times \frac{\nu_{8P}}{\ell_{22P,P}}(D_Q)$$

$f_{7,P}$ cannot be gotten using the last two representations, so the computation cannot proceed to go. In other words,

$$f_{7,P} = f_{8,P} \times \frac{\nu_{8P}}{\ell_{7P,P}}(D_Q)$$

$$f_{7,P} \neq f_{8,P} \times \frac{\nu_{8P}}{\ell_{-7P,P}}(D_Q)$$

$$f_{7,P} \neq f_{8,P} \times \frac{\nu_{8P}}{\ell_{22P,P}}(D_Q)$$

finally, $\ell_{-7P,P}$ should be $\ell_{7P,P}$.

Conclusively, the correction of *Miller's Algorithm Using Signed Digit Number* is accurate and works.

5.2 New Right-to-left Miller's Algorithm

New Right-to-left Miller's Algorithm will use two core equations:

$$f_{2m,P} = f_{m,P}^2 \times \frac{\ell_{[m]P,[m]P}}{\nu_{[2m]P}} \tag{25}$$

and

$$f_{m+n,P} = f_{m,P} \times f_{n,P} \times \frac{\ell_{[m]P,[n]P}}{\nu_{[m+n]P}} \tag{26}$$

Equation (25) has been proved in the Subsection (4.2) from the Page 28 to the Page 31. Observe that, Equation (26) just substitutes m in Equation (25) with n; moreover, it will be attested in divisor level.

- From $f_{m,P}$ and $f_{n,P}$ to $f_{m+n,P}$:

  According to Equation (16),

$$(f_{m,P}) = m(P) - ([m]P) - (m-1)(\mathcal{O}) \tag{27}$$

  substituting m with n and $m+n$, then

$$(f_{n,P}) = n(P) - ([n]P) - (n-1)(\mathcal{O}) \tag{28}$$

$$(f_{m+n,P}) = (m+n)(P) - ([m+n]P) - (m+n-1)(\mathcal{O}) \tag{29}$$

  Moreover,

$$(\ell_{[m]P,[n]P}) = ([m]P) + ([n]P) + (-[m+n]P) - 3(\mathcal{O}), \tag{30}$$

$$(\nu_{[m+n]P}) = (-[m+n]P) + ([m+n]P) - 2(\mathcal{O}), \tag{31}$$

  Therefore, in divisor level

$$Equation(29) = Equation(27) + Equation(28) + Equation(30) - Equation(31)$$

  Namely,

$$f_{m+n,P} = f_{m,P} \times f_{n,P} \times \frac{\ell_{[m]P,[n]P}}{\nu_{[m+n]P}}$$

The New Right-to-left Miller's Algorithm calculates r from the least significant digit to the most significant digit (where r is a binary number of length n), in other words, the computational sequence is from right to left, but the existing works are on the contrary, so the proposal proposed a new option. On the other hand, it can compute the $f_{r,P}(D_Q)$ more efficiently, and it will be compared with the existing works in Chapter 6.

Algorithm 5.2 New Right-to-left Miller's Algorithm

Input: $P \in E(\mathbb{F}_{q^k})[r]$, $D_Q \sim (Q) - (\mathcal{O})$ with support disjoint from $(f_{r,P})$, and $r = (r_{n-1}...r_1r_0)_2$, and $r_i \in \{0,1\}$.

Output: $f_{r,P}(D_Q) \leftarrow f_\alpha$.

```
1: $P_1 \leftarrow \mathcal{O},\ f_\alpha \leftarrow f_{1,P} = 1;\quad P_2 \leftarrow P,\ f_\beta \leftarrow f_{1,P} = 1.$
2: for $i = 0$ up to $n-1$ do
3:     if $r_i = 1$, then
4:         Compute the line functions $\ell_{P_1,P_2};\ \ell_{P_2,P_2}$.
5:         $P_1 \leftarrow P_1 + P_2;\quad P_2 \leftarrow [2]P_2$.
6:         Compute the line functions $\nu_{P_1};\ \nu_{P_2}$.
7:         $f_\alpha \leftarrow f_\alpha \times f_\beta \times \frac{\ell_{P_1,P_2}}{\nu_{P_1}}(D_Q);\quad f_\beta \leftarrow f_\beta^2 \times \frac{\ell_{P_2,P_2}}{\nu_{P_2}}(D_Q)$.
8:     else
9:         Compute the line function $\ell_{P_2,P_2}$.
10:        $P_2 \leftarrow [2]P_2$.
11:        Compute the line function $\nu_{P_2}$.
12:        $f_\beta \leftarrow f_\beta^2 \times \frac{\ell_{P_2,P_2}}{\nu_{P_2}}(D_Q)$.
13:    end if
14: end for
15: return $f_\alpha$.
```

1. There is a little trick to deal with the last loop:

   $r_i \in \{0,1\}$ means the first digit r_{n-1} must equal to 1; in addition, the final return is f_α rather than f_β. Therefore, when the Algorithm 5.2 computes the last loop ($r_{n-1} = 1$), just doing the process to compute the value of f_α is fine, and it is unnecessary to do the process to compute the value of f_β.

   In other words, let the loops be just from $i = 0$ up to $n-2$ rather than from $i = 0$ up to $n-1$, and when computing the last digit ($r_{n-1} = 1$), it computes the steps of computing the f_α and removes the steps of computing the f_β:

when $r_{n-1} = 1$,

- Compute the line function ℓ_{P_1,P_2}.
- $P_1 \leftarrow P_1 + P_2$
- Compute the line function ν_{P_1}.
- $f_\alpha \leftarrow f_\alpha \times f_\beta \times \frac{\ell_{P_1,P_2}}{\nu_{P_1}}(D_Q)$.

Consequently, when it preforms the little trick, it is capable of saving the steps of computing the f_β in the last loop.

2. An example of the New Right-to-left Miller's Algorithm:

   Input: $P,\ D_Q,\ r = 53 = (110101)_2$

   Output: $f_{53,P}(D_Q) \leftarrow f$

   Compute:

   (a) $\mathcal{O},\quad f_{1,P} = 1; \quad P,\quad f_{1,P} = 1.$

   (b) $r_0 = 1$:

      i. $\ell_{\mathcal{O},P}; \quad \ell_{P,P}$

      ii. $P = \mathcal{O} + P; \quad 2P = 2 \times P$

      iii. $\nu_P; \quad \nu_{2P}$

      iv. $f_{1,P} = f_{1,P} \times f_{1,P} \times \frac{\ell_{\mathcal{O},P}}{\nu_P}(D_Q) = 1; \quad f_{2,P} = f_{1,P}^2 \times \frac{\ell_{P,P}}{\nu_{2P}}(D_Q)$

   (c) $r_1 = 0$:

      i. $\ell_{2P,2P}$

      ii. $4P = 2 \times 2P$

      iii. ν_{4P}

      iv. $f_{4,P} = f_{2,P}^2 \times \frac{\ell_{2P,2P}}{\nu_{4P}}(D_Q)$

(d) $r_2 = 1$:

   i. $\ell_{P,4P}$; $\ell_{4P,4P}$

   ii. $5P = P + 4P$; $8P = 2 \times 4P$

   iii. ν_{5P}; ν_{8P}

   iv. $f_{5,P} = f_{1,P} \times f_{4,P} \times \frac{\ell_{P,4P}}{\nu_{5P}}(D_Q)$; $f_{8,P} = f_{4,P}^2 \times \frac{\ell_{4P,4P}}{\nu_{8P}}(D_Q)$

(e) $r_3 = 0$:

   i. $\ell_{8P,8P}$

   ii. $16P = 2 \times 8P$

   iii. ν_{16P}

   iv. $f_{16,P} = f_{8,P}^2 \times \frac{\ell_{8P,8P}}{\nu_{16P}}(D_Q)$

(f) $r_4 = 1$:

   i. $\ell_{5P,16P}$; $\ell_{16P,16P}$

   ii. $21P = 5P + 16P$; $32P = 2 \times 16P$

   iii. ν_{21P}; ν_{32P}

   iv. $f_{21,P} = f_{5,P} \times f_{16,P} \times \frac{\ell_{5P,16P}}{\nu_{21P}}(D_Q)$; $f_{32,P} = f_{16,P}^2 \times \frac{\ell_{16P,16P}}{\nu_{32P}}(D_Q)$

(g) $r_5 = 1$:

   i. $\ell_{21P,32P}$; $\ell_{32P,32P}$

   ii. $53P = 21P + 32P$; $64P = 2 \times 32P$

   iii. ν_{53P}; ν_{64P}

   iv. $f_{53,P} = f_{21,P} \times f_{32,P} \times \frac{\ell_{21P,32P}}{\nu_{53P}}(D_Q)$; $f_{64,P} = f_{32,P}^2 \times \frac{\ell_{32P,32P}}{\nu_{64P}}(D_Q)$

(h) return $f_{53,P}$

Notice:

- In step (b), according to the functions of chord line and vertical line,

$$(\ell_{\mathcal{O},P}) = (P) + (\mathcal{O}) + (-P) - 3(\mathcal{O})$$

$$(\nu_P) = (P) + (-P) - 2(\mathcal{O})$$

obtain $(\frac{\ell_{\mathcal{O},P}}{\nu_P}) = (\ell_{\mathcal{O},P} - \nu_P) =$ zero divisor, it can be $f_{1,P} = 1$.

Generally, the equation can be an universal equation,

$$(\frac{\ell_{\mathcal{O},[m]P}}{\nu_{[m]P}}) = (\ell_{\mathcal{O},[m]P} - \nu_{[m]P}) = \text{ zero divisor, it can be 1.}$$

because of

$$(\ell_{\mathcal{O},[m]P}) = ([m]P) + (\mathcal{O}) + ([-m]P) - 3(\mathcal{O})$$

$$(\nu_{[m]P}) = ([m]P) + ([-m]P) - 2(\mathcal{O})$$

Namely, when the computation needs to calculate $\ell_{\mathcal{O},[m]P}$ and $\nu_{[m]P}$, it does not have to calculate them, and can obtain the result of $\frac{\ell_{\mathcal{O},[m]P}}{\nu_{[m]P}} = 1$, directly.

- In step (g), it can use the little trick which is mentioned at the Page 44, so that it can save the computational steps of $f_{64,P}$. Thus, the step (g) could be:

  (g) $r_5 = 1$:

    i. $\ell_{21P,32P}$

    ii. $53P = 21P + 32P$

    iii. ν_{53P}

    iv. $f_{53,P} = f_{21,P} \times f_{32,P} \times \frac{\ell_{21P,32P}}{\nu_{53P}}(D_Q)$

3. The contributions of the New Right-to-left Miller's Algorithm:

   (a) It calculates the $r = (r_{n-1}...r_1r_0)_2$ with a new sequence (from right to left).

- This is a new option for some particular designs.
- It can make cyber attackers confused: even if they obtain every digits of r, they may not guess the sequence of r which is not the conventional sequence (from left to right).

  Thus, the new sequence (from right to left) may be securer.

(b) When $r_i = 1$, there is a semicolon (**;**) to separate two computations in every step, that means the separated computations are independent, and their computations can start at the same time and not influence each other. In other words, the separated computations by a semicolon are parallel computation.

Hence, when $r_i = 1$, the computational time is just the maximum of doubling stage and addition stage. Nevertheless, the existing works are all serial computation that the computational time is the sum of doubling stage and addition stage. Therefore, the New Right-to-left Miller's Algorithm speeds up and is more efficient. There will be more detailed comparison in next chapter.

5.3 Modified Miller's Algorithm with Enhanced Security

The aim of the modified Miller's Algorithm is against certain side channel attack in pairing-based cryptography (PBC), so it has to assure the complexities of two conditions (when $r_i = 1$ and $r_i = 0$) are the same, so that attackers cannot analyze out which r_i equals to 1 or 0. Thus, they are not able to obtain the final value of r. In other words, the modified Miller's Algorithm is secure in against certain side channel attack, i.e., simple power analysis.

The followings are the modified algorithm and its instance; additionally, the analyses of the complexities will be given in next chapter.

Algorithm 5.3 Modified Miller's Algorithm with Enhanced Security

Input: $P \in E(\mathbb{F}_{q^k})[r]$, $D_Q \sim (Q) - (\mathcal{O})$ with support disjoint from $(f_{r,P})$, and $r = (r_{n-1}...r_1r_0)_2$ with $r_{n-1} = 1$, and $r_i \in \{0,1\}$.

Output: $f_{r,P}(D_Q) \leftarrow f_\alpha$.

```
1: $P_1 \leftarrow P,\ \ f_\alpha \leftarrow f_{1,P} = 1;\quad P_2 \leftarrow [2]P,\ \ f_\beta \leftarrow f_{2,P} = \frac{\ell_{P,P}}{\nu_{[2]P}}(D_Q)$.
2: for $i = n-2$ down to 0 do
3:   if $r_i = 1$, then
4:     Compute the line functions $\ell_{P_1,P_2};\ \ell_{P_2,P_2}$.
5:     $P_1 \leftarrow P_1 + P_2;\quad P_2 \leftarrow [2]P_2$.
6:     Compute the line functions $\nu_{P_1};\ \nu_{P_2}$.
7:     $f_\alpha \leftarrow f_\alpha \times f_\beta \times \frac{\ell_{P_1,P_2}}{\nu_{P_1}}(D_Q);\quad f_\beta \leftarrow f_\beta^2 \times \frac{\ell_{P_2,P_2}}{\nu_{P_2}}(D_Q)$.
8:   else
9:     Compute the line functions $\ell_{P_1,P_2};\ \ell_{P_1,P_1}$.
10:    $P_2 \leftarrow P_1 + P_2;\quad P_1 \leftarrow [2]P_1$.
11:    Compute the line functions $\nu_{P_2};\ \nu_{P_1}$.
12:    $f_\beta \leftarrow f_\alpha \times f_\beta \times \frac{\ell_{P_1,P_2}}{\nu_{P_2}}(D_Q);\quad f_\alpha \leftarrow f_\alpha^2 \times \frac{\ell_{P_1,P_1}}{\nu_{P_1}}(D_Q)$.
13:  end if
14: end for
15: return $f_\alpha$.
```

1. Algorithm 5.3 still use the following two core equations:

$$f_{2m,P} = f_{m,P}^2 \times \frac{\ell_{[m]P,[m]P}}{\nu_{[2m]P}}$$

and

$$f_{m+n,P} = f_{m,P} \times f_{n,P} \times \frac{\ell_{[m]P,[n]P}}{\nu_{[m+n]P}}$$

They were proved in previous sections, the first one has been proved in the Subsection (4.2) from the Page 28 to the Page 31, and the second one has been proved in the Subsection (5.2) from the Page 42 to the Page 44.

2. An example of the Modified Miller's Algorithm with Enhanced Security:

Input: $P,\ D_Q,\ r = 53 = (110101)_2$

Output: $f_{53,P}(D_Q) \leftarrow f$

Compute:

(a) $P, \quad f_{1,P} = 1; \quad 2P, \quad f_{2,P} = \frac{\ell_{P,P}}{\nu_{[2]P}}(D_Q)$.

(b) $r_4 = 1$:

  i. $\ell_{P,2P}; \quad \ell_{2P,2P}$

  ii. $3P = P + 2P; \quad 4P = 2 \times 2P$

  iii. $\nu_{3P}; \quad \nu_{4P}$

  iv. $f_{3,P} = f_{1,P} \times f_{2,P} \times \frac{\ell_{P,2P}}{\nu_{3P}}(D_Q); \quad f_{4,P} = f_{2,P}^2 \times \frac{\ell_{2P,2P}}{\nu_{4P}}(D_Q)$

(c) $r_3 = 0$:

  i. $\ell_{3P,4P}; \quad \ell_{3P,3P}$

  ii. $7P = 3P + 4P; \quad 6P = 2 \times 3P$

  iii. $\nu_{7P}; \quad \nu_{6P}$

  iv. $f_{7,P} = f_{3,P} \times f_{4,P} \times \frac{\ell_{3P,4P}}{\nu_{7P}}(D_Q); \quad f_{6,P} = f_{3,P}^2 \times \frac{\ell_{3P,3P}}{\nu_{6P}}(D_Q)$

(d) $r_2 = 1$:

  i. $\ell_{7P,6P}; \quad \ell_{7P,7P}$

  ii. $13P = 7P + 6P; \quad 14P = 2 \times 7P$

  iii. $\nu_{13P}; \quad \nu_{14P}$

  iv. $f_{13,P} = f_{7,P} \times f_{6,P} \times \frac{\ell_{7P,6P}}{\nu_{13P}}(D_Q); \quad f_{14,P} = f_{7,P}^2 \times \frac{\ell_{7P,7P}}{\nu_{14P}}(D_Q)$

(e) $r_1 = 0$:

  i. $\ell_{13P,14P}; \quad \ell_{13P,13P}$

  ii. $27P = 13P + 14P; \quad 26P = 2 \times 13P$

  iii. $\nu_{27P}; \quad \nu_{26P}$

  iv. $f_{27,P} = f_{13,P} \times f_{14,P} \times \frac{\ell_{13P,14P}}{\nu_{27P}}(D_Q); \quad f_{26,P} = f_{13,P}^2 \times \frac{\ell_{13P,13P}}{\nu_{26P}}(D_Q)$

(f) $r_0 = 1$:

  i. $\ell_{27P,26P}; \quad \ell_{27P,27P}$

  ii. $53P = 27P + 26P; \quad 54P = 2 \times 27P$

iii. ν_{53P}; ν_{54P}

iv. $f_{53,P} = f_{27,P} \times f_{26,P} \times \frac{\ell_{27P,26P}}{\nu_{53P}}(D_Q)$; $f_{54,P} = f_{27,P}^2 \times \frac{\ell_{27P,27P}}{\nu_{54P}}(D_Q)$

(g) return $f_{53,P}$

3. The contributions of the Modified Miller's Algorithm with Enhanced Security:

   (a) As aforementioned, it is against certain side channel attack, i.e., simple power analysis because the computational complexities of two different conditions are equal; namely, no matter when $r_i = 1$ or $r_i = 0$, the computations are balance, so that the value of r cannot be analyzed out. Thus, this proposed algorithm can be against certain side channel attack.

   (b) The computational efficiency are not reduced in comparison with the existing works, even more efficient because of the parallel computation.

   In every loop, there is a semicolon (;) to separate two computations in every step, that means the separated computations are independent, and their computations can start at the same time and not impact mutually. In other words, the separated computations by a semicolon are parallel computation.

   Consequently, the computational time is just the maximum of doubling stage and addition stage. However, the existing works are serial computation which the computational time is the sum of doubling stage and addition stage. Therefore, the Modified Miller's Algorithm with Enhanced Security is not only more efficient than the existing works but also has the nice property of resistance to certain side channel attacks.

6 COMPLEXITY ANALYSIS AND COMPARISON

In this chapter, the comparison amongst the two existing works and the two proposed works will be demonstrated. Additionally, it will analyze the complexities in computational cost level. Namely,

- M: represents the computational cost of a multiplication;
- S: represents the computational cost of a squaring;
- I: represents the computational cost of an inversion.

The analyses ignore the computational cost of addition (A) because it is trivial in comparison with any of M, S and I. What's more, the computational cost $I \gg 20M$ over $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$, and "the multiplication to inversion ratio is commonly reported to be 80 : 1 or higher" [1].

6.1 Computational Complexity Analysis

6.1.1 Complexity Analysis of Points Operation over Elliptic Curve

In the Section (2.4.3) at the Page 15:

Let $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ be two points on the curve $E: \quad y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}$.

* Assume $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2) \neq \mathcal{O}$, then

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2 \\ y_3 = \lambda(x_1 - x_3) - y_1 \end{cases}$$

where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$, if $P_1 \neq P_2$; and $\lambda = \frac{3x_1^2 + a}{2y_1}$, if $P_1 = P_2$.

Observably,

- Points doubling costs: $2M + 2S + I$
- Points addition costs: $2M + S + I$

6.1.2 Complexity Analysis of Straight Lines

In the Section (2.4.5) at the Page 17:

Let elliptic curve E be given as $E: \quad y^2 = x^3 + ax + b$, and also let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on E.

- Chord line,
$$\ell_{P,Q}: \quad y - \frac{y_2 - y_1}{x_2 - x_1} \cdot x - \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1}$$
- Tangent line,
$$\ell_{P,P}: \quad y - \frac{3x_1^2 + a}{2y_1} \cdot x - \frac{-y_1^2 + 2ax_1 + 3b}{2y_1}$$
- Vertical line,
$$\nu_Q: \ x - x_2$$

Note:

It just considers the costs which are caused by x_1, y_1 and x_2, y_2, excluding the unknown coordinates x, y.

In the existing works and the proposed works, the steps of computing lines just calculate out the line FUNCTIONS; namely, they obtain the representations of x, y, and x, y do not need to be calculated. On the other hand, after it completes the steps of computing lines, substituting the values of x, y into the line functions are other steps, and at that time x just multiplies a constant. Therefore, the costs which are caused by x, y are ignored.

According to the straight line equations,

- Chord line costs: $2M + 2I$
- Tangent line costs: $2S + 2I$
- Vertical line costs: *null*

6.1.3 Complexity Analysis of $\frac{\ell}{\nu}(D_Q)$

In the point (3) of the Section (4.2) at the Page 35:

Based on

$$\frac{\ell}{\nu}(D_Q) = \frac{\ell([2]Q)}{\ell(Q)} \times \frac{\nu(Q)}{\nu([2]Q)} = \frac{\ell([2]Q) \times \nu(Q)}{\ell(Q) \times \nu([2]Q)}$$

It can be observed that both chord line and tangent line have the same cost: $2M + I$.

6.1.4 Complexity Analysis of the Existing Works and the Proposed Works

In this subsection, it will analyze the complexity of each step when $r_i = 0$, $r_i = 1$, and $r_i = \overline{1}$, respectively. In addition, they are all based on each computable step of every algorithm and the previous complexity analyses.

1. **Miller's Algorithm**: Table 6.1 analyzes the computational costs of every step when $r_i = 0$ and when $r_i = 1$.

Table 6.1: Complexity Analysis of Miller's Algorithm

Step	$r_i = 0$	$r_i = 1$
3:	$2S + 2I$	$2S + 2I$
4:	$2M + 2S + I$	$2M + 2S + I$
5:	*null*	*null*
6:	$3M + S + I$	$3M + S + I$
8:		$2M + 2I$
9:		$2M + S + I$
10:		*null*
11:		$3M + I$
Total:	$5M + 5S + 4I$	$12M + 6S + 8I$

Therefore,

- when $r_i = 0$, the totally computational cost is $5M + 5S + 4I$;
- when $r_i = 1$, the totally computational cost is $12M + 6S + 8I$.

2. **Miller's Algorithm Using Signed Digit Number**: Table 6.2 presents the computational costs of every step and the total when $r_i = 0$, $r_i = 1$, and $r_i = \overline{1}$, respectively.

Table 6.2: Complexity Analysis of Miller's Algorithm Using Signed Digit Number

Step	$r_i = 0$	$r_i = 1$	$r_i = \overline{1}$
3:	$2S + 2I$	$2S + 2I$	$2S + 2I$
4:	$2M + 2S + I$	$2M + 2S + I$	$2M + 2S + I$
5:	*null*	*null*	*null*
6:	$3M + S + I$	$3M + S + I$	$3M + S + I$
8 (14):		$2M + 2I$	*null*
9 (15):		$2M + S + I$	$2M + S + I$
10 (16):		*null*	$2M + 2I$
11 (17):		$3M + I$	$3M + I$
Total:	$5M + 5S + 4I$	$12M + 6S + 8I$	$12M + 6S + 8I$

Thus,

- when $r_i = 0$, the totally computational cost is $5M + 5S + 4I$;
- when $r_i = 1$, the totally computational cost is $12M + 6S + 8I$;
- when $r_i = \overline{1}$, the totally computational cost is $12M + 6S + 8I$.

3. **New Right-to-left Miller's Algorithm:** In Table 6.3, the computational costs of every step and the total are presented when $r_i = 0$ and when $r_i = 1$, as well as the parts of parallel computation.

Table 6.3: Complexity Analysis of New Right-to-left Miller's Algorithm

Step	$r_i = 0$	Step	$r_i = 1$	
9:	$2S + 2I$	4:	$2M + 2I$	$2S + 2I$
10:	$2M + 2S + I$	5:	$2M + S + I$	$2M + 2S + I$
11:	*null*	6:	*null*	*null*
12:	$3M + S + I$	7:	$4M + I$	$3M + S + I$
Total:	$5M + 5S + 4I$	Total:	$8M + S + 4I$	$5M + 5S + 4I$

Hence,

- when $r_i = 0$, the totally computational cost is $5M + 5S + 4I$;
- when $r_i = 1$, the totally computational cost is $(8M + S + 4I) + (5M + 5S + 4I) = 13M + 6S + 8I$.

4. **Modified Miller's Algorithm with Enhanced Security:** In Table 6.4, the computational costs of every step and the total are analyzed when $r_i = 0$ and when $r_i = 1$, as well as the parts of parallel computation.

Table 6.4: Complexity Analysis of Modified Miller's Algorithm with Enhanced Security

Step	$r_i = 0$		Step	$r_i = 1$	
9:	$2M + 2I$	$2S + 2I$	4:	$2M + 2I$	$2S + 2I$
10:	$2M + S + I$	$2M + 2S + I$	5:	$2M + S + I$	$2M + 2S + I$
11:	*null*	*null*	6:	*null*	*null*
12:	$4M + I$	$3M + S + I$	7:	$4M + I$	$3M + S + I$
Total:	$8M + S + 4I$	$5M + 5S + 4I$	Total:	$8M + S + 4I$	$5M + 5S + 4I$

Consequently,

- when $r_i = 0$, the totally computational cost is $(8M + S + 4I) + (5M + 5S + 4I) = 13M + 6S + 8I$.
- Similarly, when $r_i = 1$, the totally computational cost is $13M + 6S + 8I$.

6.2 Computational Complexity Comparison

Here uses the following names to represent the existing works and the proposed works for abbreviation.

- Original MA: Miller's Algorithm [2],

- MA using SD: Miller's Algorithm Using Signed Digit Number [3],
- New R2L MA: New Right-to-left Miller's Algorithm,
- Modified MA: Modified Miller's Algorithm with Enhanced Security

1. Complexity analysis for proposed and existing works when $r_i = 0$ are analyzed and compared in Table 6.5.

Table 6.5: Complexity Analysis when $r_i = 0$

Algorithms	Original MA	MA using SD	New R2L MA	Modified MA
Total Cost	$5M + 5S + 4I$	$5M + 5S + 4I$	$5M + 5S + 4I$	$13M + 6S + 8I$

2. When $r_i = 1$, because only Miller's Algorithm Using Signed Digit Number has the condition of $r_i = \overline{1}$, and the complexity of $r_i = \overline{1}$ equals to that of $r_i = 1$, so Table 6.6 just lists the computational complexities when $r_i = 1$.

Table 6.6: Complexity Analysis when $r_i = 1$

Algorithms	Original MA	MA using SD	New R2L MA	Modified MA
Total Cost	$12M + 6S + 8I$	$12M + 6S + 8I$	$13M + 6S + 8I$	$13M + 6S + 8I$

3. Let r be a binary number of length n.

   - Normally, in an *n-bit* binary number: the number of 0 is $\frac{n}{2}$, and the number of 1 is also $\frac{n}{2}$.
   - Usually, if we convert an *n-bit* binary number to a signed digit number, then the signed digit number will be $(n+1)$-*bit*: and the number of 0 is $\frac{2(n+1)}{3}$, and the total number of 1 and $\overline{1}$ is $\frac{(n+1)}{3}$ [40].

   Based on the four algorithms, the number of loops of every algorithm is listed in Table 6.7:

Table 6.7: Comparison: the Number of Loops

Name	Original MA	MA using SD	New R2L MA	Modified MA
# of Loop	$n-1$	n	n	$n-1$

Hence, based on Table 6.5, Table 6.6 and Table 6.7, the computational complexity comparison is obtained in Table 6.8:

Table 6.8: Computational Complexity Comparison

Name	Total Cost
Original MA	$(8.5n-8.5)M+(5.5n-5.5)S+(6n-6)I$
MA using SD	$(7.3n)M+(5.3n)S+(5.3n)I$
New R2L MA	$(9n)M+(5.5n)S+(6n)I$
Modified MA	$(13n-13)M+(6n-6)S+(8n-8)I$

Therefore, Miller's Algorithm Using Signed Digit Number [3] is the best in computational complexity comparison.

6.3 Space-time Diagrams of the Existing Works and the Proposed Works

6.3.1 Space-time Diagram of the Existing Works

When either one of the existing works completes one loop, it will need the doubling stage (step 3 to step 6) plus the addition stage (step 8 to step 11); thus, the time per loop is the sum of doubling stage time and addition stage time, and this situation could be called serial computation. The details can be illustrated in Fig. 6.1:

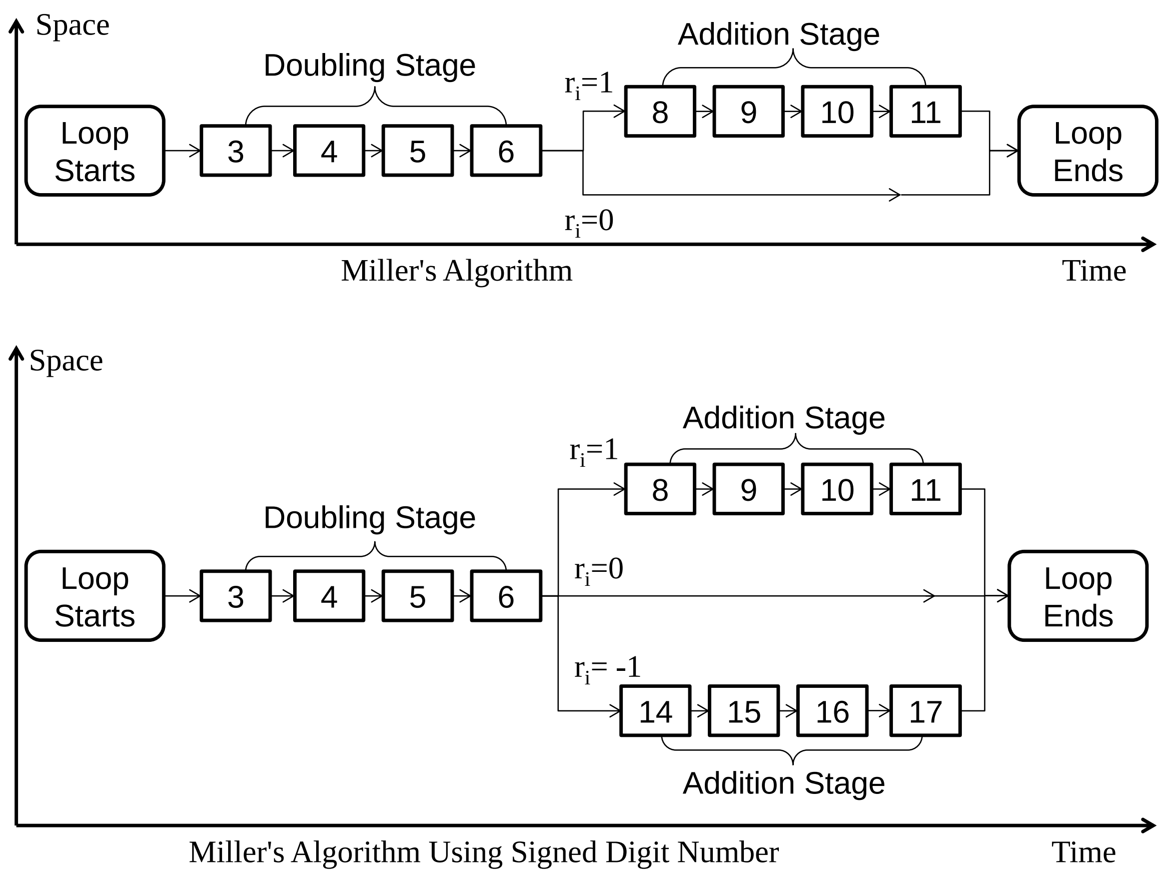
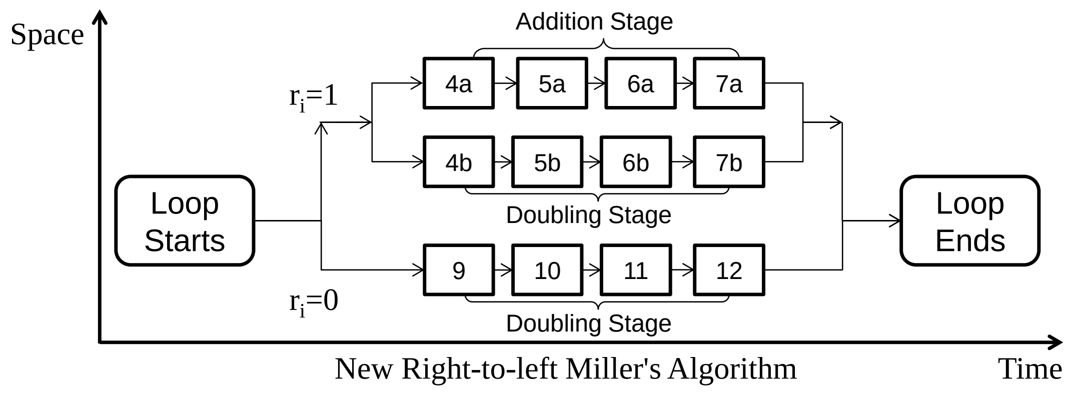


Fig. 6.1: Space-time Diagram of the Existing Works

6.3.2 Space-time Diagram of the Proposed Works

On the other hand, when either one of the proposed works completes one loop, the addition stage (step 4a to step 7a) and the doubling stage (step 4b to step 7b) can start simultaneously and approximately finish at the same time. In other words, the two stages are independent mutually, and cannot influence each other, and this situation could be called parallel computation. The details can be illustrated in Fig. 6.2.

As a result, the computational time is the worse case of doubling stage time and addition stage time. According to the complexity analyses, the addition stage complexity $8M + S + 4I \approx$ the doubling stage complexity $5M + 5S + 4I$, that means the computational time of the two stages are approximately equated, so the time per loop can be either one of the doubling stage time and the addition stage time.



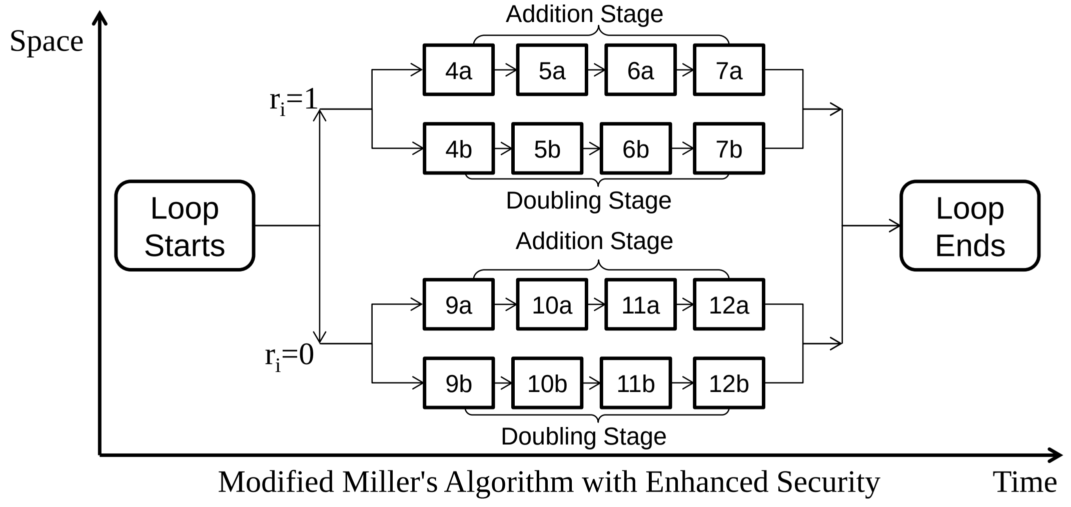


Fig. 6.2: Space-time Diagram of the Proposed Works

Note: Each of steps 4, 5, 6, 7, 9, 10, 11 and 12 can be split into two parallel sub-steps:

For clearly showing the different parts of step 4 to step 12, the diagram uses the 4a to 12a to represent the addition stage, and the 4b to 12b to represent the doubling stage.

In the proposed works, the semicolon (;) separates the “Series ‘a’ steps” and the “Series ‘b’ steps” . For instance,

---

4: Compute the line functions (4a:) ℓ_{P_1,P_2}; (4b:) ℓ_{P_2,P_2} in parallel.

```
5: (5a:) $P_1 \leftarrow P_1 + P_2$;   (5b:) $P_2 \leftarrow [2]P_2$ in parallel.
6: Compute the line functions (6a:) $\nu_{P_1}$;  (6b:) $\nu_{P_2}$ in parallel.
7: (7a:) $f_\alpha \leftarrow f_\alpha \times f_\beta \times \frac{\ell_{P_1,P_2}}{\nu_{P_1}}(D_Q)$;   (7b:) $f_\beta \leftarrow f_\beta^2 \times \frac{\ell_{P_2,P_2}}{\nu_{P_2}}(D_Q)$ in parallel.
```

6.4 Performance Comparison

A comparison of the proposed algorithms and the existing works is shown in Table 6.9. Note that T_A and T_D denote the time delay for addition stage and doubling stage, respectively.

Table 6.9: Comparison: Proposed Works vs. Existing Works

Algorithms	Total Cost	Loop delay	Latency	Input style (r)	SCA resistance
Original MA	$(8.5n-8.5)M+(5.5n-5.5)S+(6n-6)I$	T_A+T_D	$(n-1)(T_A+T_D)$	Left-to-right	No
MA using SD	$(7.3n)M+(5.3n)S+(5.3n)I$	T_A+T_D	nT_A+nT_D	Left-to-right	No
New R2L MA	$(9n)M+(5.5n)S+(6n)I$	T_A	nT_A	Right-to-left	No
Modified MA	$(13n-13)M+(6n-6)S+(8n-8)I$	T_A	$(n-1)T_A$	Left-to-right	Yes

The comparison results shown in Table 6.9 can be explained as follows.

1. The proposed works contain parallelable computation steps, while the existing works are serial computation. As a consequence, assumed parallel implementation for the proposed works, then it can realize high speed computation of Weil pairing.

2. Where r is a binary number of length n, according to the security level, the value of n is from 224 to 512.

3. The loop delay of the proposed works is less than that of the existing works.

4. The latency of the proposed works is less than that of the existing works.

5. The difference of the two proposed works:

    - In the aspect of input style, New Right-to-left Miller's Algorithm proposes the Right-to-Left, which is a new option for design;

    - Although the latency of New Right-to-left Miller's Algorithm is more a T_A than that of Modified Miller's Algorithm with Enhanced Security, the $(n-1)T_A$ is very big, so it can be deemed the latency of the two proposed works are approximately equal;

    - In New Right-to-left Miller's Algorithm, only when $r_i = 1$, it uses the parallel computation; on the contrary, in Modified Miller's Algorithm with Enhanced Security, no matter when $r_i = 1$ or $r_i = 0$, they both use the parallel computation, so that the fourth algorithm needs more space to implement.

Conclusively, the proposed works are superior to the existing works; in addition, comparing with the two proposed works, they have individual and irreplaceable advantage.

7 CONCLUSIONS

7.1 Research Contributions and Applications

After the first chapter on introduction, the thesis provides the mathematical preliminaries of the pairing-based cryptography, including finite fields, elliptic curve, and the important concepts of divisors and bilinear map, followed by an overview of existing works on Miller's Algorithm. Then the two modified versions of Miller's Algorithm are proposed. An elaborate comparison between the proposed works and existing works in complexity and performance is also presented.

The research contributions presented in this thesis include the followings,

- Two new algorithms for computing bilinear map are presented. The first one is the New Right-to-left Miller's Algorithm and the second is called the Modified Miller's Algorithm with Enhanced Security.
- It is clear that the first proposed algorithm works when the input r is fed into system in right-to-left fashion while the original Miller's takes input only in left-to-right style.
- Both the proposed algorithms possess parallelism within each loop and thus make it possible for parallel and high speed computation, compared to the original Miller's. Elaborated analytical results show that the improvement in speed could be close to 100 percent if parallel implementation is ensured.
- The second proposed algorithm has the property of resistance to certain side channel attacks, i.e., simple power analysis, which makes it a better choice for computing bilinear map when there exists potential threat of side channel attacks.

7.2 Possible Future Works

Based on the research works proposed in this thesis, the following research directions may be worthy of further investigation:

- Hardware implementation, i.e., FPGA implementation, of the proposed algorithms for computing bilinear map, which can maximally take advantage of parallelism in the algorithms and their computational efficiency.
- It is expected that the proposed algorithms can be easily extended for computation of Tate pairing.

REFERENCES

[1] C. Costello, “Pairings for beginners,” www.craigcostello.com.au/pairings/ PairingsForBeginners.pdf, 2012.

[2] V. S. Miller, “The weil pairing, and its efficient calculation,” *Journal of Cryptology*, vol. 17, no. 4, pp. 235–261, 2004.

[3] X. Zhang and D. Lin, “Analysis of optimum pairing products at high security levels,” in *International Conference on Cryptology in India.* Springer, 2012, pp. 412–430.

[4] A. Menezes, “An introduction to pairing-based cryptography,” *Recent trends in cryptography*, vol. 477, pp. 47–65, 2009.

[5] “ ieee standard for identity-based cryptographic techniques using pairings,” *IEEE Std 1363.3-2013*, pp. 1–151, Nov 2013.

[6] A. Joux, “A one round protocol for tripartite diffie–hellman,” *Journal of cryptology*, vol. 17, no. 4, pp. 263–276, 2004.

[7] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *Annual International Cryptology Conference.* Springer, 2001, pp. 213–229.

[8] R. Sakai, K. Ohgishi, and M. Kasahara, “Cryptosystems based on pairing, scis 2000-c20, jan. 2000,” *Okinawa, Japan.*

[9] A. Joux, “A one round protocol for tripartite diffie–hellman,” in *International Algorithmic Number Theory Symposium.* Springer, 2000, pp. 385–393.

[10] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in *International Conference on the Theory and Application of Cryptology and Information Security.* Springer, 2001, pp. 514–532.

[11] C. Zhang, X. Lin, R. Lu, P.-H. Ho, and X. Shen, "An efficient message authentication scheme for vehicular communications," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 6, pp. 3357–3368, 2008.

[12] T. Schütze, "Automotive security: Cryptography for car2x communication," in *Embedded World Conference*, vol. 3, 2011.

[13] S. Lee, Y. Kim, K. Kim, and D.-H. Ryu, "An efficient tree-based group key agreement using bilinear map," in *International Conference on Applied Cryptography and Network Security.* Springer, 2003, pp. 357–371.

[14] A. Kim, V. Kniss, G. Ritter, and S. M. Sloan, "An approach to communications security for a communications data delivery system for v2v/v2i safety: Technical description and identification of policy and institutional issues," Tech. Rep., 2011.

[15] W. Whyte, A. Weimerskirch, V. Kumar, and T. Hehn, "A security credential management system for v2v communications," in *Vehicular Networking Conference (VNC), 2013 IEEE.* IEEE, 2013, pp. 1–8.

[16] K. Benson, H. Shacham, and B. Waters, "The k-bdh assumption family: Bilinear map cryptography from progressively weaker assumptions," in *Cryptographers Track at the RSA Conference.* Springer, 2013, pp. 310–325.

[17] R. Lu, "Security and privacy preservation in vehicular social networks," 2012.

[18] R. Dutta, R. Barua, and P. Sarkar, "Pairing-based cryptography: A survey," *Cryptology ePrint Archive, Report 2004/064*, 2004.

[19] A. Kumano and Y. Nogami, "An improvement of tate paring with supersingular curve," in *Information Science and Security (ICISS), 2015 2nd International Conference on.* IEEE, 2015, pp. 1–3.

[20] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the Theory and Application of Cryptographic Techniques.* Springer, 1985, pp. 417–426.

[21] V. Miller *et al.*, "Short programs for functions on curves," *Unpublished manuscript*, vol. 97, pp. 101–102, 1986.

[22] H. Wu, "Lecture notes in Data Security," September 2015.

[23] R. Murty and I. Shparlinski, "Group structure of elliptic curves over finite fields and applications," in *Topics in Geometry, Coding Theory and Cryptography.* Springer, 2006, pp. 167–194.

[24] S. Gashkov, A. Frolov, and I. Sergeev, "Arithmetic in finite fields supporting type-2 or type-3 optimal normal bases," in *Dependability Engineering and Complex Systems.* Springer, 2016, pp. 157–168.

[25] H. Niederreiter, H. Wang, and C. Xing, "Function fields over finite fields and their applications to cryptography," in *Topics in Geometry, Coding Theory and Cryptography.* Springer, 2006, pp. 59–104.

[26] M. F. De Oliveira and M. A. A. Henriques, "A secure and efficient method for scalar multiplication on supersingular elliptic curves over binary fields," in *Information Security.* Springer, 2015, pp. 407–416.

[27] Y.-T. Chang, Y.-H. Liu, and R.-J. Chen, "Selecting elliptic curves with minimal estimation of pairing computation," in *The Second International Conference on Information Security and Digital Forensics (ISDF2015)*, 2015, p. 21.

[28] G. Frey and T. Lange, *Mathematical background of public key cryptography.* IEM, 2003.

[29] M. Scott and P. S. Barreto, "Generating more mnt elliptic curves," *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 209–217, 2006.

[30] J. M. Miret, D. Sadornil, and J. Tena, "Elliptic curves with j= 0, 1728 and low embedding degree," *International Journal of Computer Mathematics*, vol. 93, no. 12, pp. 2042–2053, 2016.

[31] R. Schoof, "Elliptic curves over finite fields and the computation of square roots mod q," *Mathematics of computation*, vol. 44, no. 170, pp. 483–494, 1985.

[32] S. D. Galbraith, *Mathematics of public key cryptography.* Cambridge University Press, 2012.

[33] T. Okamoto, "Cryptography based on bilinear maps," in *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes.* Springer, 2006, pp. 35–50.

[34] T. Yamakawa, S. Yamada, G. Hanaoka, and N. Kunihiro, "Self-bilinear map on unknown order groups from indistinguishability obfuscation and its applications," in *International Cryptology Conference.* Springer, 2014, pp. 90–107.

[35] N. El Mrabet, "What about vulnerability to a fault attack of the millers algorithm during an identity based protocol?" in *International Conference on Information Security and Assurance.* Springer, 2009, pp. 122–134.

[36] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[37] A. Joux and K. Nguyen, "Separating decision diffie–hellman from computational diffie–hellman in cryptographic groups," *Journal of cryptology*, vol. 16, no. 4, pp. 239–247, 2003.

[38] P.-A. Fouque and C. Qian, "Fault attacks on efficient pairing implementations," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security.* ACM, 2016, pp. 641–650.

[39] I. F. Blake, V. K. Murty, and G. Xu, "Refinements of miller's algorithm for computing the weil/tate pairing," *Journal of Algorithms*, vol. 58, no. 2, pp. 134–149, 2006.

[40] H. Wu and M. A. Hasan, "Efficient exponentiation of a primitive root in gf (2/sup m/)," *IEEE Transactions on Computers*, vol. 46, no. 2, pp. 162–172, 1997.

[41] D.-P. Le and C.-L. Liu, "Refinements of miller's algorithm over weierstrass curves revisited," *The Computer Journal*, vol. 54, no. 10, pp. 1582–1591, 2011.

[42] F. Vercauteren, "Optimal pairings," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 455–461, 2010.

VITA AUCTORIS

NAME:	Shun Wang
PLACE OF BIRTH:	Shenyang, China
YEAR OF BIRTH:	1985
EDUCATION:	
2004 - 2008	Harbin University of Science and Technology, Harbin, Heilongjiang, China Bachelor of Electrical and Electronic Engineering
2014 - 2017	University of Windsor, Windsor, Ontario, Canada Master of Applied Science, Electrical and Computer Engineering
EMAIL:	solomonshun@gmail.com