
Chapter 1

Introduction

1.1 Introduction to Cryptography

In recent years the fast development of the communication and network, brings the people to be quick and make a convenience of life, but the data its safety is very important while delivering not burglarize to take the exploitation, and use cryptography to protect the data is the most familiar method.

Recently, public key cryptography (PKC) have received more and more attention. It is more and more important in digital communication and some data transfer systems such as home banking, internet, electronic commerce, E-mail that are needed to be kept secret from insecure channel. According to the difficult mathematical problem on which they are based, there are three types of systems classified and are thought secure [8] :

1. Integer factorization systems (RSA).
2. Discrete logarithm systems (U.S. Government's DSA).
3. Elliptic curve discrete logarithm systems (Elliptic Curve cryptography).

In 1976 Diffie and Hellman [1] introduced a concept of a public key cryptography and described a public key distribution scheme. The security of this proposed concept is based on the difficult and intractable discrete logarithm problem

in the multiplicative group of a large prime finite field. In 1977 the complete cryptography was firstly proposed by three researchers at MIT, Rivest, Shamir and Adleman [3]. The famous system is called RSA cryptography and is believed that its security is based on factoring in a large integer. In 1985, ElGamal [2] proposed a practical public key cryptography based on discrete exponentiation problem in a finite field.

Some compromises are required between system response time and security. For current cryptography, they are up to required security standards but computational speed is always compromised due to increased key size. As we have seen, the bit length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA. Recently, a competing system has begun to challenge RSA : elliptic curve cryptography (ECC).

Elliptic curves have been studied for over one hundred fifty years. However, until 1985 elliptic curve public key cryptography was first proposed by Victor Miller [4] and Neal Koblitz [5]. Already, ECC is showing up in standardization efforts, including the IEEE P1363 Standard for Public-Key Cryptography.

There are some applications such as smart cards and mobile phones which are portable and small device used in many ways like identification, and health care needed cryptographic service. But there are some restrictions on using these devices : limited computing power bandwidth and constrained memory. To handle these problems, we can choose the advantages of ECC which has smaller key size and high security. Hence, ECC permits reductions in key and delivers the highest strength per bit of any known public-key system because of the elliptic curve discrete logarithm problem (ECDLP). We can compare the key length between ECC and RSA system on

the same security level and then get a result showing that the smaller key size of ECC yield equivalent levels of security of RSA system. Therefore, the elliptic curve cryptography can be thought as one of the best public key cryptography in the world today.

Many hardware architectures have been proposed [6],[7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] for elliptic curve cryptography (ECC) [19], [20]. Binary field $GF(2^m)$ arithmetic is suitable for fast and compact hardware compared with a prime field $GF(p)$ because there is no discrimination between positive and negative numbers and, thus, no carry is propagated. However, conventional implementations for $GF(2^m)$ ECC cannot support EC-DSA [22] over $GF(2^m)$, which is one of the most important ECC standard functions because modular arithmetic in $GF(p)$ is also required. On the other hand, conventional ECC hardware designs in $GF(p)$ [17], [18] supported only the specific prime modulus $p=2^{192}-2^{64}-1$. Therefore, we proposed a scalable architecture for dual-field $GF(p)$ and $GF(2^m)$ based on Montgomery Modular Multiplication.

1.2 Organization of this thesis

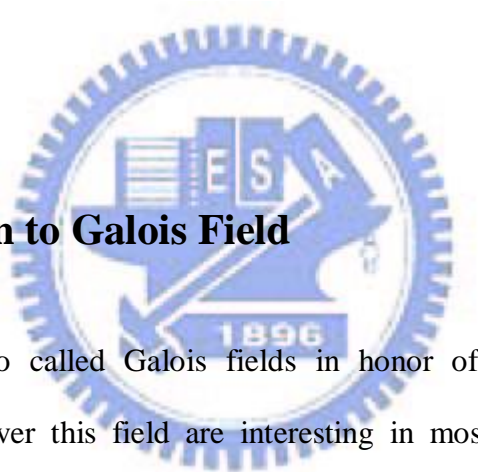
In Chapter 2, we will introduce some mathematical background : finite fields, Fermat's theorem and Euclid's Algorithm. In Chapter 3, we will discuss the ECC and mathematical fundamentals of elliptic curve. In Chapter 4, some protocols such as analog of ElGamal Public Key Cryptosystem, ECDH, ECDSA, and some standards are presented. In Chapter 5, we will present our modified architecture and simulation results. Finally, conclusions will be given in Chapter 6.

Chapter 2

Mathematical Background

This chapter introduces some mathematical background for Elliptic Curve. In Section 2.1, we introduce the Galois Field. And we will introduce Fermat's Theorem and Euclid's Algorithm which are usually used to find the multiplicative inversion in Section 2.2.

2.1 Introduction to Galois Field



Finite field is also called Galois fields in honor of its discoverer and all arithmetic operations over this field are interesting in most computer engineering domain included cryptography. A finite field in an algebraic field that has a finite number of elements, otherwise called Infinite Field.

Due to there are finite numbers of elements on Galois Field, any operation on an element of the field will result in another one in this field. Because of this useful property, algorithm using finite field arithmetic does not need to cope with over or under flow problem.

2.1.1 Finite field $GF(p)$

The set $\{0,1,2,\dots,p-1\}$ is a field of order p under modulo- p addition and

multiplication. Since this field is constructed from a prime p , it is called a *prime field* and is denoted by $GF(p)$. For any prime p , there exists a finite field of p elements. In fact, for any positive integer m , it is possible to extend the prime field $GF(p)$ to a field of p^m elements which is called an *extension field* of $GF(p)$ and is denoted by $GF(p^m)$. Furthermore, it has been proved that the order of any finite field is a power of a prime.

In Table 2.1 and Table 2.2, we show the addition and multiplication operation in $GF(7)$, respectively.

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

Table 2.1 Modulo 7 addition.

x	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Table 2.2 Modulo 7 multiplication.

2.1.2 Finite field $GF(2^m)$

Arithmetic on $GF(2^m)$ is less complexity than traditional binary. On the other hand, $GF(2^m)$ arithmetic is suited to hardware implementation. Addition and subtraction is simply an XOR operation. Multiplication is more difficult, but can be implemented by using pure combinational logic. This makes $GF(2^m)$ arithmetic can be easily implemented by ASICs or FPGAs.

To construct the $GF(2^m)$, we need to know what is irreducible polynomial and primitive polynomial.

For a polynomial $f(X)$ over $GF(2)$, if it has an even number of terms, it is divisible by $X+1$. A polynomial $p(X)$ over $GF(2)$ of degree of m is said to be irreducible over $GF(2)$ if $p(X)$ is not divisible by any polynomial over $GF(2)$ of degree less than m but greater than zero. For example, polynomials of degree of 2, X^2 , X^2+1 and X^2+X are not irreducible because they are either divisible X or $X+1$. For X^2+X+1 as an example, this polynomial does not have “0” or “1” as a root and so is not divisible by any polynomial of degree 1.

A irreducible polynomial $p(X)$ of degree m is said to be primitive if the smallest positive integer n for which $p(X)$ divides X^n+1 is $n=2^m-1$. For instance, $p(X)=X^4+X+1$ divides $X^{15}+1$ but does not divide any X^n+1 for $1 \leq n \leq 15$. Therefore, X^4+X+1 is a primitive polynomial.

An example is given below to show how to construct the $GF(2^m)$.

Example :

To construct the finite field $GF(2^4)$, firstly we choose a primitive polynomial

$p(X) = X^4 + X + 1$ which is over $GF(2)$. Secondly, set $p(a) = 1 + a + a^4 = 0$ which implies that $a^4 = 1 + a$. Using that, we can construct $GF(2^4)$, and elements on which are given in Table 2-3. The identity $a^4 = 1 + a$ is also used repeatedly to produce the polynomial representations for the elements of $GF(2^4)$. For example,

$$\begin{aligned} a^5 &= a \cdot a^4 = a(1+a) = a + a^2, \\ a^6 &= a \cdot a^5 = a(a + a^2) = a^2 + a^3, \\ a^7 &= a \cdot a^6 = a(a^2 + a^3) = a^3 + a^4 \\ &= a^3 + 1 + a = 1 + a + a^3 \end{aligned}$$

To multiply two elements a^j and a^i , we simply add their exponents and use the fact that $a^{15} = 1$. For example, $a^5 \cdot a^6 = a^{11}$ and $a^8 \cdot a^{14} = a^{22} = a^7 \cdot a^{15} = a^7$. To divide a^j by a^i , we simply multiply a^j by multiplicative inverse a^{15-i} of a^i . For example, $\frac{a^5}{a^{14}} = a^5 \cdot a = a^7$. To add a^j and a^i , we simply use their polynomial representations in Table 2.3. For example,

$$\begin{aligned} a^9 + a^7 &= (a + a^3) + (1 + a + a^3) = 1 \\ 1 + a^3 + a^{11} &= 1 + a^3 + (a + a^2 + a^3) = 1 + a + a^2 = a^{10} \end{aligned}$$

There is another representation called m -tuple representation for $GF(2^m)$. The components of the m -tuple representation are the coefficients of the polynomial representation. All three representations are given in Table 2.3.

Power representation	Polynomial representation	4-Tuple or binary representation
0	0	(0 0 0 0)
1	1	(1 0 0 0)
a	a	(0 1 0 0)
a^2	a^2	(0 0 1 0)
a^3	a^3	(0 0 0 1)

a^4	$1+a$	$(1\ 1\ 0\ 0)$
a^5	$a+a^2$	$(0\ 1\ 1\ 0)$
a^6	a^2+a^3	$(0\ 0\ 1\ 1)$
a^7	$1+a+a^3$	$(1\ 1\ 0\ 1)$
a^8	$1+a^2$	$(1\ 0\ 1\ 0)$
a^9	$a+a^3$	$(0\ 1\ 0\ 1)$
a^{10}	$1+a+a^2$	$(1\ 1\ 1\ 0)$
a^{11}	$a+a^2+a^3$	$(0\ 1\ 1\ 1)$
a^{12}	$1+a+a^2+a^3$	$(1\ 1\ 1\ 1)$
a^{13}	$1+a^2+a^3$	$(1\ 0\ 1\ 1)$
a^{14}	$1+a^3$	$(1\ 0\ 0\ 1)$

Table 2.3 Three representations for the elements of $GF(2^4)$ generated by $p(x)=X^4+X+1$ [29].

2.2 Fermat's Theorem and Euclid's Algorithm

When we want to find a multiplicative inverse modulo p , we can use Fermat's theorem or Euclid's Algorithm. Finding a multiplicative inverse is an important step in public-key cryptography, especially in elliptic curve cryptography. Therefore, in this section, we will introduce these two usefully methods.

2.2.1 Fermat's theorem

Fermat's theorem states the following :

If p is prime and a is a positive integer not divisible by p , then

$$a^{p-1} \equiv 1 \pmod{p} \quad (2.1)$$

Proof :

We know that if all the elements of Z_p are multiplied by a , modulo p , the

result consists of the elements of Z_p in some order. Furthermore, $a \times 0 \equiv 0 \pmod p$.

Therefore, the $(p-1)$ numbers $\{a \pmod p, 2a \pmod p, \dots, (p-1)a \pmod p\}$ are just the numbers $\{1, 2, \dots, (p-1)\}$ in some order. Multiply these numbers together :

$$\begin{aligned} a \times 2a \times \dots \times (p-1)a &\equiv [(a \pmod p) \times (2a \pmod p) \times \dots \times (p-1)a \pmod p] \pmod p \\ &\equiv (p-1)! \pmod p \end{aligned} \quad (2.2)$$

But

$$a \times 2a \times \dots \times ((p-1)a) \equiv (p-1)! a^{p-1} \quad (2.3)$$

We can cancel the $(p-1)!$ term both in (7.2) and (7.3) because it is relatively prime to p . This yields Equation (2.1).

From Equation (2.1), we can easily derive the Equation

$$a^{-1} \equiv a^{p-2} \pmod p. \quad (2.4)$$

So, the multiplicative inverse of a modulo p can be find from Fermat's theorem.

2.2.2 Euclid's algorithm

One of the basic techniques of number theory is Euclid's algorithm, which is a simple procedure for determining the greatest common divisor of two positive integers. An extended form of Euclid's algorithm determines the greatest common divisor of two positive integers and, if those numbers are relatively prime, the multiplicative inverse of one with respect to the other.

Euclid's algorithm is based on the following theorem : For any nonnegative integer a and any positive integer b ,

$$\gcd(a, b) = \gcd(b, a \pmod b) \quad (2.5)$$

Euclid's algorithm makes repeated use of Equation (2.5) to determine the

greatest common divisor, as follows :

Assume $f > d > 0$. It is acceptable to restrict the algorithm to positive integers because $\gcd(a, b) = \gcd(|a|, |b|)$.

EUCLID (d, f)

1. $X \leftarrow d; Y \leftarrow f$
2. if $Y = 0$ return $X = \gcd(d, f)$
3. $R = X \bmod Y$
4. $X \leftarrow Y$
5. $Y \leftarrow R$
6. goto 2

If $\gcd(d, f) = 1$, then d has a multiplicative inverse modulo f . That is for positive integer $d < f$, there exists a $d^{-1} < f$ such that $dd^{-1} = 1 \bmod f$. Euclid's algorithm can be extended so that, in addition to finding $\gcd(d, f)$, if the gcd is 1, the algorithm returns the multiplicative inverse of d .

EXTENDED EUCLID (d, f)

1. $(X1, X2, X3) \leftarrow (1, 0, f); (Y1, Y2, Y3) \leftarrow (0, 1, d)$
2. if $Y3 = 0$ return $X3 = \gcd(d, f)$; no inverse
3. if $Y3 = 1$ return $Y3 = \gcd(d, f); Y2 = d^{-1} \bmod f$
4. $Q = \left\lfloor \frac{X3}{Y3} \right\rfloor$
5. $(T1, T2, T3) \leftarrow (X1 - QY1, X2 - QY2, X3 - QY3)$
6. $(X1, X2, X3) \leftarrow (Y1, Y2, Y3)$
7. $(Y1, Y2, Y3) \leftarrow (T1, T2, T3)$
8. goto 2

Throughout the computation, the following relationship hold :

$$fT1 + dT2 = T3 \quad fX1 + dX2 = X3 \quad fY1 + dY2 = Y3$$

Note that if $\gcd(d, f) = 1$, then on the final step we would have $Y3 = 0$ and $X3 = 1$. Therefore, on the preceding step, $Y3 = 1$. But if $Y3 = 1$, then we can say the following :

$$fY_1 + dY_2 = Y_3$$

$$fY_1 + dY_2 = 1$$

$$dY_2 = 1 + (-Y_1) \times f$$

$$dY_2 \equiv 1 \pmod{f}$$

And Y_2 is the multiplicative inverse of d , modulo f .

Table 2.4 is an example of the execution of the algorithm. It shows that $\gcd(550, 1769) = 1$ and that the multiplicative inverse of 550 is itself; that is, $550 \times 550 = 1 \pmod{1769}$.

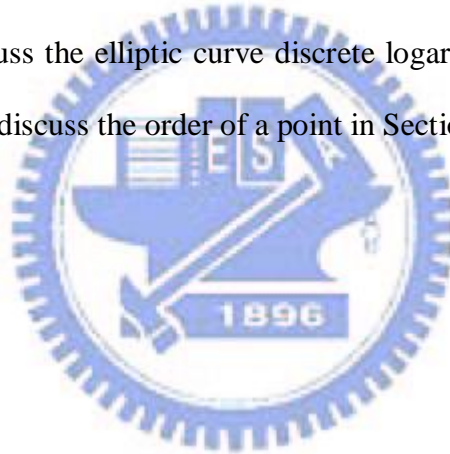
Q	X1	X2	X3	Y1	Y2	Y3
	1	0	1769	0	1	550
3	0	1	550	1	-3	119
4	1	-3	119	-4	13	74
1	-4	13	74	5	-16	45
1	5	-16	45	-9	29	29
1	-9	29	29	14	-45	16
1	14	-45	16	-23	74	13
1	-23	74	13	37	-119	3
4	37	-119	3	-171	550	1

Table 2.4 An example of Extended Euclid (550, 1769).

Chapter 3

Overview of Elliptic Curves

We introduce an overview of Elliptic curve Cryptography in this chapter. There are a large amount of papers proposed on this subject. In Section 3.1, we introduce the history of ECC. Basic theories used in ECC are introduced in Section 3.2. Arithmetic of ECC is discussed in Section 3.3. In Section 3.4, we will discuss the Hasse's theorem. Then, we discuss the elliptic curve discrete logarithm problem (ECDLP) in Section 3.5. Finally, we discuss the order of a point in Section 3.6.



3.1 History

Elliptic curves have been studied intensively for the past 150 years and there are a large amount of papers proposed on this subject. Elliptic curves have appeared a rich and deep theory. Moreover, the discrete logarithm problem (DLP) is believed to be very difficult. Elliptic curve systems were first suggested in 1985 independently by Neal Koblitz [5] at the University of Washington, Victor Miller [4], and Yorktown Heights for implementing public key cryptosystem.

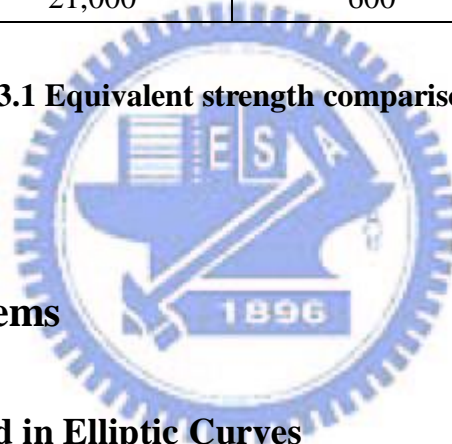
ECC offers a solution for those public-key systems which need most constrained environments such as smaller size, faster computing. This is the reason that ECC is a good scheme for low memory, low bandwidth, and low power consumption

applications such as smart cards and mobile communication.

The difficulty of the elliptic curve discrete logarithm problem (ECDLP) means that smaller key sizes provide equivalent levels of security. Table 3.1 shows that equivalent strength comparison between RSA/DSA and ECC.

Time to break In MIPS years	RSA/DSA key size (bits)	ECC key size (bits)	RSA/ECC Key size ratio
10^4	512	106	5 : 1
10^8	768	132	6 : 1
10^{11}	1,024	160	7 : 1
10^{20}	2,048	210	10 : 1
10^{78}	21,000	600	35 : 1

Table 3.1 Equivalent strength comparison.



3.2 Basic theorems

3.2.1 Theorems used in Elliptic Curves

Let F_q denote the finite field containing q elements, where q is a prime power. If $K = F_q$, let \bar{k} denote its algebraic closure, i.e., $\bar{k} = \bigcup_{m \geq 1} F_q^m$. A projective plane over K is the set of an equivalent class where if and only if there exists $l \in K^*$ satisfying $(x_1, y_1, z_1) = (l x_2, l y_2, l z_2)$. Let the projective plane $P^2(K)$ over K be the set of $(x : y : z) \setminus \{(0 : 0 : 0)\}$ in K^3 . We denote projective points (x, y, z) on $P^2(K)$ by (x, y, z) . We will describe a special equation called Weierstrass equation which is a homogeneous equation of degree 3 of the form :

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

where $a_1, a_2, a_3, a_4, a_6 \in \overline{K}$. An elliptic curve E is the set of points satisfying a nonsingular Weierstrass equation with a special point O . There is a special point called the point at infinity has Z -coordinate equal to 0 denoted by O , namely $(0,1,0)$. The Weierstrass equation is said to be non-singular if for all projective points $Q = (x : y : z) \in P^2(\overline{K})$ satisfying

$$F(X, Y, Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 = 0,$$

and at least one of the three partial derivatives $\frac{\partial F}{\partial X}, \frac{\partial F}{\partial Y}, \frac{\partial F}{\partial Z}$ is not zero at Q .

By using affine coordinates $x = X/Z, y = Y/Z$, we can derive the Weierstrass equation for an elliptic curve :

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

and then an elliptic curve E is the set of solutions of above equation in the affine plane K^2 , together with the point at infinity O . If $a_1, a_2, a_3, a_4, a_6 \in K$, E is said to be defined over K denoted by E/K , and the set of points both of whose coordinates lie in K , together with the point O is called K -rational points of E , denoted $E(K)$.

Theorem 3.1 [26,p16] Two elliptic curves E_1/K and E_2/K given by the equations

$$\begin{aligned} E_1 : y^2 + a_1xy + a_3y &= x^3 + a_2x^2 + a_4x + a_6 \\ E_2 : y^2 + \overline{a_1}xy + \overline{a_3}y &= x^3 + \overline{a_2}x^2 + \overline{a_4}x + \overline{a_6} \end{aligned}$$

are isomorphic over K , denoted $E_1/K \cong E_2/K$, If and only if there exists $u, r, s, t \in K, u \neq 0$, such that the change of variables

$$(x, y) \rightarrow (u^2x + r, u^3y + u^2sx + t)$$

transforms equation E_1 to equation E_2 . The relationship of isomorphism is an equivalence relation.

Theorem 3.2 [26,p17] Two elliptic curves E_1/K and E_2/K are isomorphic over K if and only if there exist $u, r, s, t \in K, u \neq 0$ satisfying

$$\begin{cases} u\bar{a}_1 = a_1 + 2s \\ u^2\bar{a}_2 = a_2 - sa_1 + 3r - s^2 \\ u^3\bar{a}_3 = a_3 + ra_1 + 2t \\ u^4\bar{a}_4 = a_4 - sa_3 + 2ra_2 - (t + rs)a_1 + 3r^2 - 2st \\ u^6\bar{a}_6 = a_6 + ra_4 + r^2a_2 + r^3 - ta_3 - t^2 - rta_1 \end{cases}$$

The Discriminant and i-Invariant

Let us define the quantities of elliptic curve given by affine Weierstrass equation.

$$\begin{cases} d_2 = a_1^2 + 4a_2 \\ d_4 = 2a_4 + a_1a_3 \\ d_6 = a_3^2 + 4a_6 \\ d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \\ c_4 = d_2^2 - 24d_4 \\ \Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6 \\ j(E) = c_4^3 / \Delta \end{cases} \quad (3-2)$$

The quantity Δ defined above is called the discriminant of the Weierstrass equation. We call $j(E)$ j -invariant of E if $\Delta \neq 0$. Significance of these quantities is described in the following two theorems.

Theorem 3.2 [26,p17] E is an elliptic curve, i.e., the Weierstrass equation is non-singular, if and only if $\Delta \neq 0$.

Theorem 3.2 [26,p17] If two elliptic curves E_1/K and E_2/K are isomorphic over K , then $j(E_1) = j(E_2)$. The converse is also true if K is an algebraically closed field.

3.2.2 Group Law

Let E be an elliptic curve given by the Weierstrass equation. For all points on an elliptic curve, we define a certain addition, denote “+”, the operation of the abelian group E/K . The addition rules are given below :

For all $P, Q \in E$	
1	$O + P = P$ and $P + O = P$
2	$-O = O$
3	If $P = (x_1, y_1) \neq O$, then $-P = (x_1, -y_1 - a_1x_1 - a_3)$
4	If $Q = -P$, then $P + Q = O$
5	If $P \neq O$, $Q \neq O$, $Q \neq -P$, then the point $-R$ is the intersection of the curve with either the line \overline{PQ} if $P \neq Q$, or the tangent line to the curve at P if $P = Q$. We define $P + Q = R$. If $P \neq Q$, we call the operation point addition. If $P = Q$, the operation is called point doubling.

Figure 3.1 is an example of point addition.

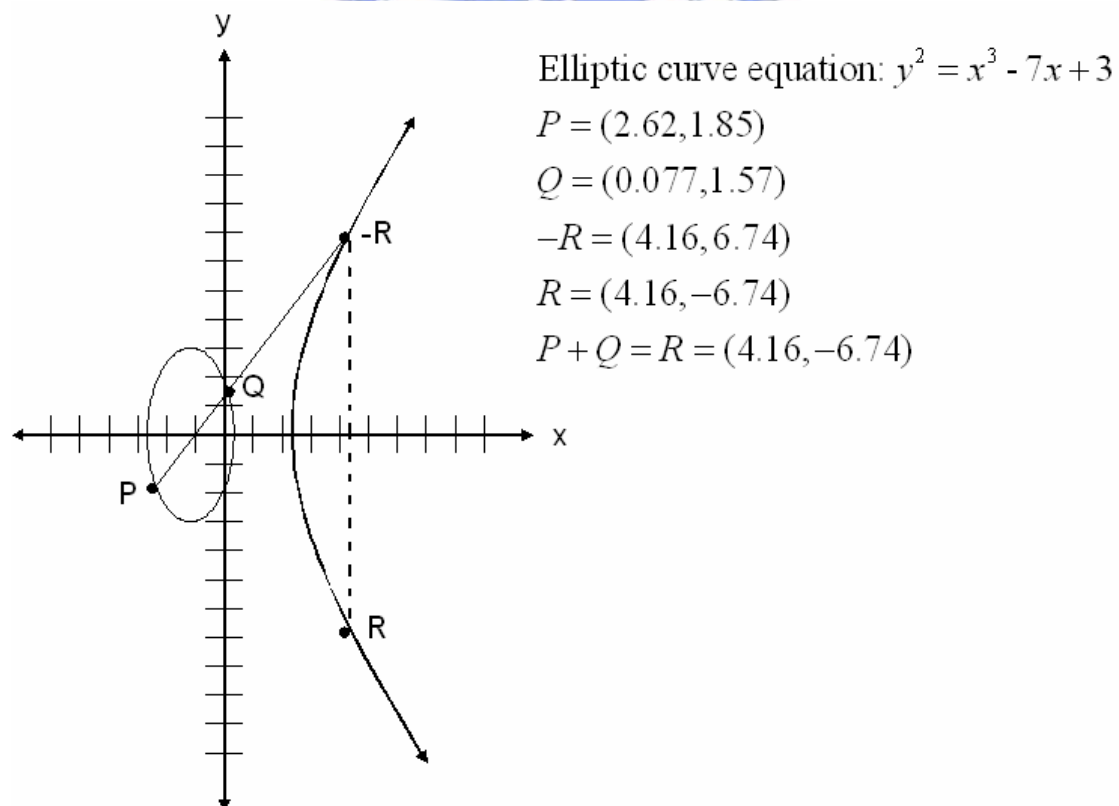


Figure 3-1 Point addition.

Curves over K of characteristic $\neq 2, 3$

Let E/K be an elliptic curve given by equation (3-1). If $\text{char}(K) \neq 2$, then we can transform E/K to the curve

$$E/K : y^2 = x^3 + b_2x^2 + b_4x + b_6$$

by using the admissible change of variables

$$(x, y) \rightarrow \left[x, y - \frac{a_1}{2}x - \frac{a_3}{2} \right].$$

Notice that $E \cong E'$ over K .

If $\text{char}(K) \neq 2, 3$, we can further transform E' to

$$E''/K : y^2 = x^3 + b_4x + b_6$$

by using the following admissible change of variables,

$$\left[\frac{x-12b}{36}, \frac{y}{216} \right].$$

Hence $E' \cong E''$ over K .

By specializing equation (3-2), we can find the associated quantities that are

$$\Delta = -16(4a^3 + 27b^2)$$

and

$$j(E) = -1728(4a)^3 / \Delta.$$

Because E is assumed to be non-singular, we can get $\Delta \neq 0$. Thus, we can get theorem 3.5 as follow by specializing theorem 3.2.

Theorem 3.5 [26, p21] The elliptic curves $E_1/K : y^2 = x^3 + ax + b$ and

$E_2/K : y^2 = x^3 + \bar{a}x + \bar{b}$ are isomorphic over K if and only if there exists a $u \in K^*$,

such that $u^4\bar{a} = a$ and $u^6\bar{b} = b$.

Assuming that $R = P + Q$, if $P = (x_1, y_1)$, $Q = (x_2, y_2)$, and $R = (x_3, y_3)$, then the addition formulas of elliptic curves over K are given below if $\text{char}(K) \neq 2, 3$.

$$\begin{aligned} x_3 &= I^2 - x_1 - x_2 \\ y_3 &= I(x_1 - x_3) - y_1 \end{aligned}$$

$$\text{where } I = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}$$

We can also find that $-P = (x_1, -y_1)$.

There are two examples showing addition operation (Example 3-1) and doubling operation of two points in the elliptic curve (Example 3-2), respectively.

Example 3-1 : Given $P = (6, 10)$ and $Q = (14, 20) \in E(F_{23}) : y^2 = x^3 + 9x + 14$, then calculating the point $R = (x_R, y_R)$, where $R = P + Q$. There are 19 solutions and one infinity point O . These points on E are

$$E = \{O, (1, 1), (1, 22), (5, 0), (6, 10), (6, 13), (7, 11), (7, 12), (8, 0), (10, 0), (11, 8), (11, 15), (14, 3), (14, 20), (19, 11), (19, 12), (20, 11), (20, 12), (22, 2), (22, 21)\}$$

Answer :

At first, calculating the slope I of \overline{PQ} , where

$$\begin{aligned} I &= (y_P - y_Q) \times (x_P - x_Q)^{-1} \pmod{P} \\ &= -10 \times (-8)^{-1} \pmod{23} \\ &= 13 \times 15^{-1} \pmod{23} \\ &= 13 \times 20 \pmod{23} \\ &= 7 \end{aligned}$$

and then calculating the x-coordinate and y-coordinate of R respectively.

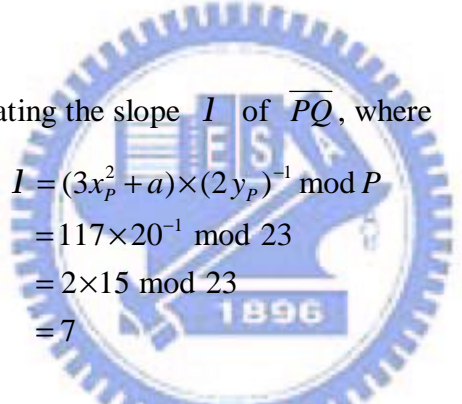
$$\begin{aligned}
x_R &= 1 - x_P - x_Q \pmod{P} \\
&= 49 - 6 - 14 \pmod{23} \\
&= 6 \\
y_R &= -y_P + I \times (x_P - x_R) \pmod{P} \\
&= -10 + 7 \times (6 - 6) \pmod{23} \\
&= 13 + 7 \times 0 \pmod{23} \\
&= 13
\end{aligned}$$

So, the answer of $R+Q$ is $R = (6,13)$.

Example 3-2 : Given $P = (6,10) \in E(F_{23})$: $y^2 = x^3 + 9x + 14$, then calculating the point $R = (x_R, y_R)$, where $R = 2P$.

Answer :

Similarly, at first, calculating the slope I of \overline{PQ} , where



$$\begin{aligned}
I &= (3x_P^2 + a) \times (2y_P)^{-1} \pmod{P} \\
&= 117 \times 20^{-1} \pmod{23} \\
&= 2 \times 15 \pmod{23} \\
&= 7
\end{aligned}$$

and then calculating the x-coordinate and y-coordinate of R respectively.

$$\begin{aligned}
y_R &= -y_P + I \times (x_P - x_R) \pmod{P} \\
&= -10 + 7 \times (6 - 14) \pmod{23} \\
&= 13 + 7 \times 15 \pmod{23} \\
&= 13 + 13 \pmod{23} \\
&= 3
\end{aligned}$$

Curves over K of characteristic 2

Now, we will discuss the elliptic curves over K of characteristic 2. We denoted $\text{char}(K)$ as characteristic of the field K . We can find that $j(E)a_1^{12} / \Delta$ by specializing equation (3-2). If $j(E) \neq 0$, then from Theorem 3.1, we can find the admissible change of variables

$$(x, y) \rightarrow \left[a_1^2 x + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3} \right]$$

transforming E to the curve

$$E_1 / K : y^2 + xy = x^3 + a_2 x^2 + a_6.$$

For E_1 , $\Delta = a_6$ and $j(E_1) = 1/a_6$.

If $j(E_1) = 0$, then the admissible change of variables

$$(x, y) \rightarrow (x + a_2, y)$$

transforming E to the curve

$$E_2 / K : y^2 + a_3 y = x^3 + a_4 x^2 + a_6$$

For E_2 , $\Delta = a_3^4$ and $j(E_2) = 0$.

If $P = (x_1, y_1)$; then $-P = (x_1, y_1 + x_1)$. If $Q = (x_2, y_2)$, and $R = (x_3, y_3)$ where $P = P + Q$, then the addition formulas of elliptic curves over K are given below if $\text{char}(K) = 2$.

If $j(E) \neq 0$,

$$x_3 = \begin{cases} \left[\frac{y_2 + y_1}{x_2 + x_1} \right]^2 + \frac{y_2 + y_1}{x_2 + x_1} + x_1 + x_2 + a_2 & , P \neq Q \\ a_1^2 + \frac{a_6}{x_1^2} & , P = Q \end{cases}$$

$$y_3 = \begin{cases} \left[\frac{y_2 + y_1}{x_2 + x_1} \right] (x_1 + x_3) + x_3 + y_1 & , P \neq Q \\ x_1^2 + \left[x_1 + \frac{y_1}{x_1} \right] x_3 + x_3 & , P = Q \end{cases}$$

If $j(E) = 0$,

$$x_3 = \begin{cases} \left[\frac{y_2 + y_1}{x_2 + x_1} \right]^2 + x_1 + x_2 & , P \neq Q \\ \frac{x_1^4 + a_4^2}{a_3^2} & , P = Q \end{cases}$$

$$y_3 = \begin{cases} \left[\frac{y_2 + y_1}{x_2 + x_1} \right] (x_1 + x_3) + y_1 + a_3 & , P \neq Q \\ \left[\frac{x_1^2 + a_4}{a_3} \right] (x_1 + x_3) + y_1 + a_3 & , P = Q \end{cases}$$

Let us take the elliptic curve $E(F_{2^4}): y^2 + xy = x^3 + g^2x^2 + g^8$ for example, there are 15 solutions and one infinity point O . These points on E are

$$E = \{O, (1,1), (1,0), (g^2,1), (g^2, g^8), (g^4, g^5), (g^4, g^8), (g^6, g^8), (g^6, g^{10}), (g^7, g^3), (g^7, g^4), (g^{12}, g^4), (g^{12}, g^6), (g^{13}, g^2), (g^{13}, g^4), (0, g^4)\}$$

and the polynomial representation used are shown in Table 2.3.

There are two examples showing the addition operation (Example 3-3) and the doubling operation (Example 3-4), respectively below.

Example 3-3: Given $P = (g^4, g^5)$ and $Q = (g^6, g^{10}) \in E(F_{2^4}): y^2 + xy = x^3 + g^2x^2 + g^8$, then calculating the point $R = (x_R, y_R)$, where $R = P + Q$.

Answer :

At first, calculating the slope I of \overline{PQ} , where

$$\begin{aligned} I &= (y_P + y_Q) \times (x_P + x_Q)^{-1} \\ &= (g^5 + g^{10}) \times (g^4 + g^6)^{-1} \\ &= 1 \times g^{-12} \\ &= g^3 \end{aligned}$$

and then calculating the x-coordinate and y-coordinate of R respectively.

$$\begin{aligned}
 x_R &= I^2 + I + x_P + x_Q + a_2 \\
 &= g^6 + g^3 + g^4 + g^6 + g^2 \\
 &= g^4
 \end{aligned}$$

$$\begin{aligned}
 y_R &= I \times (x_P + x_R) + x_R + y_P \\
 &= g^3 \times (g^4 + g^{12}) + g^{12} + g^5 \\
 &= g^3 \times g^6 + g^{12} + g^5 \\
 &= g^4
 \end{aligned}$$

So, the answer of $P+Q$ is $R = (g^{12}, g^4)$.

Example 3-4: Given $P = (g^7, g^4) \in E(F_{2^4})$: $y^2 + xy = x^3 + g^2x^2 + g^8$ then calculating the point $R = (x_R, y_R)$, where $R = 2P$.

Answer :

At first, calculating the slope I of \overline{PQ} , where

$$\begin{aligned}
 I &= x_P + y_P \times x_P^{-1} \\
 &= g^7 + g^4 \times g^{-7} \\
 &= g^7 \times g^{12} \\
 &= g^2
 \end{aligned}$$

and then calculating the x-coordinate and y-coordinate of R respectively.

$$\begin{aligned}
 x_R &= I^2 + I + a_2 \\
 &= g^4 + g^2 + g^2 \\
 &= g^4
 \end{aligned}$$

$$\begin{aligned}
 y_R &= x_P^2 + (I + 1) \times x_R \\
 &= g^{14} + (g^2 + 1) \times g^4 \\
 &= g^{14} + g^8 \times g^4 \\
 &= g^5
 \end{aligned}$$

So, the answer of $2P$ is $R = (g^4, g^5)$.

3.3 Projective Space

In affine coordinate system, we can see that adding two distinct points P and Q , where $P \neq Q$, for example, on a non-singular curve over $K = F_{2^m}$, takes two field multiplications, one square and one inversion for the addition formula. There are two field multiplication, two squares and one inversion for the doubling formula. However there are special techniques for computing inverses in F_{2^m} , a field inversion is still far more expensive than a field multiplication. There are other kinds of coordinate systems such as projective coordinate system using field multiplications instead of field inversion to speed up the operation.

3.3.1 Adding two point on elliptic curve over F_{2^m}

A non-supersingular elliptic curve E defined over F_{2^m} is an equation :

$$y^2 + xy = x^3 + a_2x^2 + a_6$$

where $a_2, a_6 \in F_{2^m}$ and $a_6 \neq 0$. Assume $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, and $P_1 \neq -P_2$. The sum $P_3 = (x_3, y_3) = P_1 + P_2$ is computed as follows :

If $P_1 \neq P_2$,

$$I = \frac{y_2 + y_1}{x_2 + x_1},$$

$$x_3 = I^2 + I + x_1 + x_2 + a_2,$$

$$y_3 = I(x_1 + x_3) + x_3 + y_1.$$

If $P_1 = P_2$,

$$I = \frac{y_1}{x_1} + x_1,$$

$$x_3 = I^2 + I + a_2,$$

$$y_3 = I(x_1 + x_3) + x_3 + y_1.$$

In either case, the computation requires two general multiplications, a squaring, and a field inversion, denoted by $2M + 1S + 1I$. Actual implementation of the elliptic curves indicates that the field addition is much faster than the field multiplication, while field inversion is more expensive than the multiplication. Therefore, the projective coordinates have been suggested to replace the inverse operation by multiplications, where a projective point (X, Y, Z) , $Z \neq 0$, maps to the affine point $(X/Z, Y/Z^2)$ [27] :

Let $P_1 = (X_1/Z_1, Y_1/Z_1^2)$ and $P_2 = (X_2/Z_2, Y_2/Z_2^2)$ be two points on the elliptic curve E . If $Z_1 = 1$, then the addition formula is $P_1 + P_2 = (X_3/Z_3, Y_3/Z_3^2)$, where

$$\begin{aligned} U &= Z_2^2 Y_1 + Y_2, S = Z_2 X_1 + X_2, T = Z_2 S \\ Z_3 &= T^2, V = Z_3 X_1, C = (X_1 + Y_1) \\ X_3 &= U^2 + T(U + S^2 + Ta_2) \\ Y_3 &= (V + X_3)(TU + Z_3) + Z_3^2 C \end{aligned}$$

Proof :

$$\begin{aligned} \frac{X_3}{Z_3} &= \frac{U^2 + T(U + S^2 + Ta_2)}{T^2} = \frac{U^2}{Z_2^2 S^2} + \frac{U}{Z_2 S} + \frac{S}{Z_2} + a_2 \\ &= \frac{(Y_1 + Y_2/Z_2^2)^2}{(X_1 + X_2/Z_2)^2} + \frac{(Y_1 + Y_2/Z_2^2)}{(X_1 + X_2/Z_2)} + \frac{X_2}{Z_2} + X_1 + a_2 \\ &= \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \left(\frac{y_1 + y_2}{x_1 + x_2}\right) + x_2 + x_1 + a_2 = x_3 \\ \frac{Y_3}{Z_3^2} &= \frac{(V + X_3)(TU + Z_3) + Z_3^2(Y_1 + X_1)}{T^4} \\ &= \frac{UT(Z_3 X_1 + X_3) + Z_3(Z_3 Y_1 + X_3)}{T^4} \\ &= \frac{(Z_2^2 Y_1 + Y_2)(Z_3 X_1 + X_3)}{T^3} + \frac{X_3}{T^2} + Y_1 \\ &= \frac{(Y_1 + Y_2/Z_2^2)}{(X_1 + X_2/Z_2)} (X_1 + X_3/Z_3) + \frac{X_3}{Z_3} + Y_1 \\ &= \left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1 = y_3 \end{aligned}$$

The number of field multiplications is 9 and the number of squarings is 5. If $a_2 = 0$ or 1, then eight general field multiplications are required.

$$\begin{aligned} \text{If } P_1 &= P_2 \\ Z_3 &= Z_1^2 X_1^2 \\ X_3 &= X_1^4 + a_6 Z_1^4 \\ Y_3 &= a_6 Z_1^4 Z_3 + X_3(a_2 Z_3 + Y_1^2 + a_6 Z_1^4) \end{aligned}$$

3.3.2 Adding two point on elliptic curve over F_q

A elliptic curve E defined over F_q is an equation :

$$y^2 = x^3 + a_4 x + a_6$$

where $a_4, a_6 \in F_q$. Assume $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, and $P_1 \neq -P_2$. The sum $P_3 = (x_3, y_3) = P_1 + P_2$ is computed as follows :

$$\begin{aligned} x_3 &\equiv I^2 - x_1 - x_2 \pmod{p} \\ y_3 &\equiv I(x_1 - x_3) - y_1 \pmod{p} \end{aligned}$$

where

$$I = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a_4}{2y_1} & \text{if } P = Q \end{cases}$$

Similar to Subsection 3.3.1, the projective coordinates have been suggested to replace the inverse operation by multiplications, where a projective point (X, Y, Z) , $Z \neq 0$, maps to the affine point $(X/Z, Y/Z^2)$.

Let $P_1 = (X_1/Z_1, Y_1/Z_1^2)$ and $P_2 = (X_2/Z_2, Y_2/Z_2^2)$ be two points on the elliptic curve E . If $Z_1 = 1$, then the addition formula is $P_1 + P_2 = (X_3/Z_3, Y_3/Z_3^2)$, where

$$A = X_1Z_2, B = Z_2^2, D = X_2 + A, E = X_2 - A, \\ F = Y_2 - Y_1B, X_3 = F^2 - Z_2DE^2, Z_3 = BE^2, \\ Y_3 = FZ_2E(X_1Z_3 - X_3) - Y_1Z_3^2$$

If $P_1 = P_2$

$$A = Z_1^2, B = X_1^2, C = 3B + a_4A, D = Y_1^2, \\ E = X_1Z_3 - X_3Z_1, Z_3 = 4Z_1D, \\ X_3 = Z_1C^2 - 8X_1D, \\ Y_3 = 2Y_1CE - 16D^2Y_1$$

Proof (case $P_1 = P_2$) :

$$\frac{X_3}{Z_3} = \frac{Z_1(3X_1^2 + a_4Z_1^2)^2 - 8X_1Y_1^2}{4Z_1Y_1^2} \\ = \left(\frac{3X_1^2 + a_4Z_1^2}{2Y_1} \right)^2 - 2\frac{X_1}{Z_1} \\ = \left(\frac{3\frac{X_1^2}{Z_1^2} + a_4}{2\frac{Y_1}{Z_1^2}} \right)^2 - 2\frac{X_1}{Z_1} \\ = \left(\frac{3x_1 + a_4}{2y_1} \right)^2 - 2x_1$$

$$\frac{Y_3}{Z_3^2} = \frac{2Y_1(3X_1^2 + a_4Z_1^2)(X_1Z_3 - X_3Z_1) - 16Y_1^5}{Z_3(4Z_1Y_1^2)} \\ = \frac{3X_1^2 + a_4Z_1^2}{2Y_1} \cdot \frac{\frac{X_1}{Z_1}Z_3 - X_3}{Z_3} - \frac{16Y_1^5}{16Z_1^2Y_1^4} \\ = \left(\frac{3\frac{X_1^2}{Z_1^2} + a_4}{2\frac{Y_1}{Z_1^2}} \right) \left(\frac{X_1}{Z_1} - \frac{X_3}{Z_3} \right) - \frac{Y_1}{Z_1^2} \\ = \left(\frac{3x_1^2 + a_4}{2y_1} \right) (x_1 - x_3) - y_1$$

3.3.3 Summary

There are several kinds of projective coordinates. In this section, we propose some kinds and compare them with affine coordinates. In the following Table 3.2, I denotes the field inversion, M denotes the field multiplication and S denotes the field square.

Coordinates		Affine Coordinate	Projective Coordinate ($X/Z, Y/Z$)	Projective Coordinate ($X/Z, Y/Z^2$)	Projective Coordinate ($X/Z^2, Y/Z^3$)
F_q	Adding	$1I+3M$	$15M$	$14M$	$16M$
	Doubling	$1I+4M$	$12M$	$14M$	$10M$
F_{2^m}	Adding	$1I+2M+1S$	$13M+1S$	$9M+5S$	$15M+5S$
	Doubling	$1I+2M+1S$	$7M+5S$	$5M+5S$	$5M+5S$

Table 3.2 Comparison of different coordinates.

3.4 The Elliptic Curve Group Structure

In this section, we will discuss the Hasse's theorem which can let us pick points P randomly and uniformly on an elliptic curve $E(F_q)$ in probabilistic polynomial time.

Let $q = p^m$, where p (a prime) is the characteristic of F_q and then let E be an elliptic curve defined over F_q . We denote $\#E(F_q)$ as the number of points in $E(F_q)$.

Because for each choice of $x \in F_q$ having at most two solutions in Weierstrass equation, we know that $\#E(F_q) < 2q+1$. For each choice of $x \in F_q$, if the

probability of having a solution of this equation in F_q is $1/2$, and we would expect

$\#E(F_q) \approx q$. The following Hasse's theorem confirms above assumption.

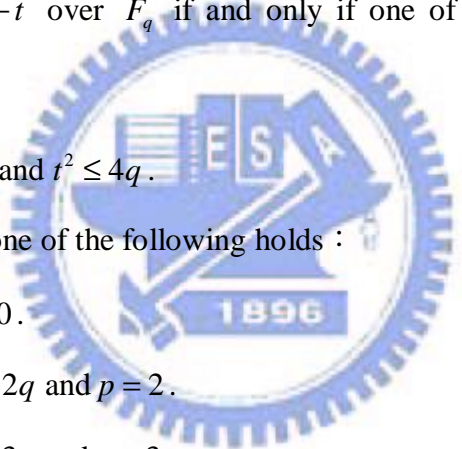
Theorem 3.6 [26, p23] (Hasse) Let $\#E(F_q) = q + 1 - t$. Then $|t| \leq 2\sqrt{q}$.

In the next result, the possible values for $\#E(F_q)$ are determined as E varies over all elliptic curves defined over F_q , where $q = p^m$.

Lemma 3.7 [26, p24] (Waterhouse) There exists an elliptic curve E/F_q such that

$E(F_q)$ has order $q + 1 - t$ over F_q if and only if one of the following conditions

holds :

- 
- i. $t \not\equiv 0 \pmod{p}$ and $t^2 \leq 4q$.
 - ii. m is odd and one of the following holds :
 - (1). $t = 0$.
 - (2). $t^2 = 2q$ and $p = 2$.
 - (3). $t^2 = 3q$ and $p = 3$
 - iii. m is even and one of the following holds :
 - (1). $t^2 = 4q$.
 - (2). $t^2 = q$ and $p \not\equiv 1 \pmod{3}$.
 - (3). $t = 0$ and $p \not\equiv 1 \pmod{4}$.

Lemma 3.7 was provided by Waterhouse. When q is prime, the values $\#E(F_q) = q + 1 - t$ will be uniformly distributed in the interval of $[q + 1 - \sqrt{q}, q + 1 + \sqrt{q}]$ which is centered at $p + 1$ when E varies over all elliptic

curves over F_q .

If p divides t then the elliptic curve E is called supersingular, where $\#E(F_q) = q+1-t$. Otherwise, it is said to be non-supersingular. It is well-known that if $p=2$ or if $p=3$, then E is supersingular curve is determined by the next two theorems.

Theorem 3.8 [26, 925] $E(F_q)$ is an abelian group of rank 1 or 2. The type of the group is (n_1, n_2) , i.e., $E(F_q) \cong Z_{n_1} \oplus Z_{n_2}$, where $n_2 | n_1$, and furthermore $n_2 | q-1$.

Lemma 3.9 [26, p25] Let $\#E(F_q) = q+1-t$.

- i. If $t^2 = q, 2q$, or $3q$, then $E(F_q)$ is cyclic.
- ii. If $t^2 = 4q$, then either $E(F_q) \cong Z_{\sqrt{q-1}} \oplus Z_{\sqrt{q-1}}$ or $E(F_q) \cong Z_{\sqrt{q+1}} \oplus Z_{\sqrt{q+1}}$, depending on whether $t = 2\sqrt{q}$ or $t = -2\sqrt{q}$ respectively.
- iii. If $t=0$ and $q \not\equiv 3 \pmod{4}$, then $E(F_q)$ is cyclic. If $t=0$ and $q \equiv 3 \pmod{4}$, then either $E(F_q)$ is cyclic, or $E(F_q) \cong Z_{(q+1)/2} \oplus Z_2$.

3.5 The Elliptic Curve Discrete Logarithm Problem

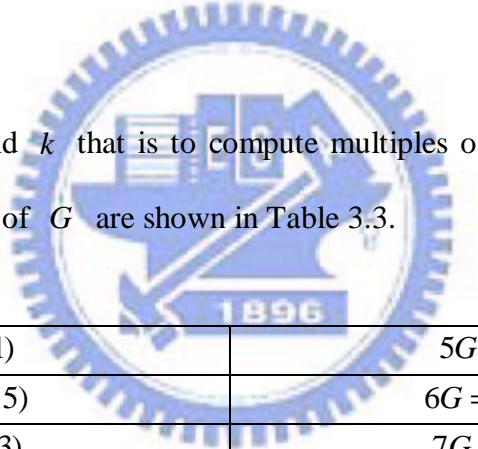
There are many public key cryptosystems whose security rely on the basis of the presumed intractability of the discrete logarithm problem in some group G .

The discrete logarithm problem in the multiplicative group Z_p^* is that given elements q and r of the group, and a prime p , find a number k such that $r = qk \pmod p$. If E is an elliptic curve over F_q , and P is the point of the elliptic curve E , then the elliptic curve discrete logarithm problem (ECDLP) is that given a point $Q \in E$ finding an integer k such that $kP = Q$ where k is called the discrete logarithm of Q to the base point P if such an integer k exists.

Example 3-5 : Let the elliptic curve $E(F_{23}) : y^2 = x^3 + 9x + 14$, what is the discrete logarithm k of $Q = (11, 8)$ to the generator point $G = (7, 11)$?

Answer :

There is one way to find k that is to compute multiples of G until Q is found. The first eight multiples of G are shown in Table 3.3.



$G = (7, 11)$	$5G = (8, 0)$
$2G = (11, 15)$	$6G = (14, 20)$
$3G = (6, 13)$	$7G = (6, 10)$
$4G = (14, 3)$	$8G = (11, 8) = Q$

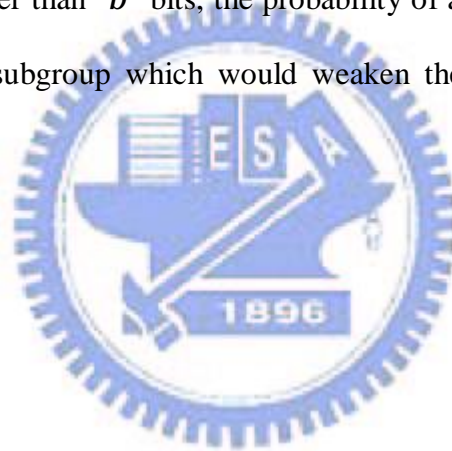
Table 3.3 The first eight multiples of generator point P .

From Table 3.3, we can see that $8G = (11, 8) = Q$, so the discrete logarithm of Q to the generator G is $k = 8$. It seems easy to find k , but it is important that in a real application, k would be large enough such that it would be very hard to determine k in this method.

3.6 The order of a point

Besides the elliptic curve discrete logarithm problem, the order of a point in an elliptic curve is also important especially the generator point. For each point $G \in E$, there is a positive integer k such that $kG \in O$. The smallest positive integer is called the order of G .

If all the factors of $\#E(F_q)$ are smaller than 2^b where b is a security parameter, then the elliptic curve is considered to be weak. Otherwise, when $\#E(F_q)$ has an prime factor larger than b bits, the probability of a generator point chosen at random yield a small subgroup which would weaken the scheme is proven to be negligible [28].



Chapter 4

Elliptic Curve Cryptography (ECC)

Now, we will describe some protocols or algorithms used in cryptography based on elliptic curve. Therefore, we need to construct an elliptic curve if we want to use EC-based cryptosystem. There are some parameters that we need to find first such as the finite field (F_q or F_{2^m}), the coefficients (a or b) of elliptic curve equation derived in chapter three, the order of $E(F_q)$ ($\#E(F_q)$ or n), and the base point of $E(F_q)$ ($P=(x,y)$).

There were three steps proposed in [23] to select an elliptic curve.

1. Let us select a curve at random, compute its order directly, and repeat the process until an appropriate order is found [24].
2. If $q = 2^m$ where m is divisible by a “small” integer, we can find curve via subfield of F_{2^m} [23].
3. Let us search for appropriate order, and construct a curve of the order. The approach is implemented using the complex multiplication method. Over a prime order field F_q , the complex multiplication method is also called the Atkin-Morain method.

4.1 Analog of ElGamal Public Key Cryptosystem

The analog of ElGamal public key cryptography as shown in Table 4.1 is one of the popular cryptography, because the elliptic curve can be changed periodically to provide much security and this is no patent protected.

As other ECCs, at first, it has to select finite F_q , an elliptic curve E define over that field and a generator point $G \in E$. Every user chooses a random integer a (d_{s_s} or d_{s_e}), which is kept a secret key by users, and computes the point $x = aG(d_{p_s}$ or $d_{p_e})$ where x is the public key.

Protocol :

Alice	insecure channel	Bob
Key Generation		Key Generation
1. Get private secret key d_{s_s} 2. Public key calculates $d_{p_s} = d_{s_s} \cdot G$		1. Get private secret key d_{s_e} 2. Public key calculates $d_{p_e} = d_{s_e} \cdot G$
Encryption	d_{p_e}	Encryption
1. Choose randomly k 2. Calculates following points: $(x_2, y_2) = k \cdot d_{p_e}$ $(x_1, y_1) = k \cdot G$ 3. Encodes message pair (m_1, m_2) with (x_2, y_2) : Encode message (c_1, c_2)	(x_1, y_1, c_1, c_2)	
Decryption		Decryption
		1. Calculates following points: $(x_2, y_2) = d_{s_e} \cdot (x_1, y_1)$ 2. (c_1, c_2) divided by (x_1, y_2) finds (m_1, m_2) : decoded message (m_1, m_2) .

Table 4.1 Analog of the ElGamal cryptosystem.

To send a message pair (m_1, m_2) to Bob, Alice choose a random integer k and calculates $k \cdot B$ and $k \cdot d_{p_e}$ (where d_{p_e} is Bob's public key). To read the message, Bob multiplies (x_1, y_1) by d_{s_e} and divides (c_1, c_2) by (x_2, y_2) .

4.2 Elliptic curve Diffie-Hellman key exchange (ECDH)

When we want to communicate in an insecure channel between two parties, a secret key is needed to achieve a safe communication. The protocol [1] of Diffie-Hellman key exchange was first proposed to allow the agreement on a secret key between two parties communicating over an insecure channel. If two communication parties, Alice and Bob, want to agree upon a key which will be later used for encrypted communication in conjunction with a symmetric key cryptosystem such as DES, Triple DES, etc..., and they first choose an elliptic curve E over a finite field publicly and a generator point $G \in E$. The important criterion in selecting G is that the smallest value on n for which $nG = O$ be a very large prime number (high order).

A key exchange between users Alice and Bob can be accomplished as follows :

1. Alice select an integer n_A less than n . This is Alice's private key. Alice then generates a public key $P_A = n_A \times G$; the public key is a point in E .
2. Bob similarly selects a private key n_B and computes a public key P_B .
3. Alice generates the secret key $K = n_A \times P_B$. Bob generates the secret key $K = n_B \times P_A$.

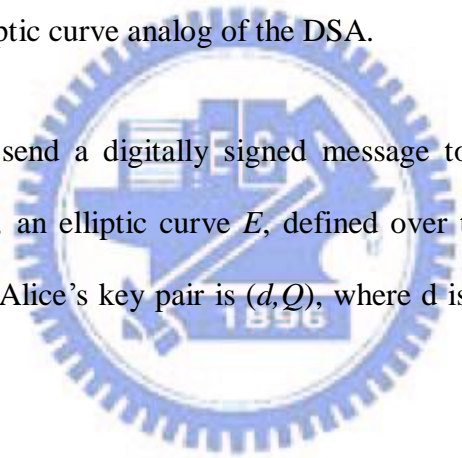
The two calculations in step 3 produce the same result because

$$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A.$$

4.3 Elliptic Curve Digital Signature Algorithm (ECDSA)

A digital signature is a number which is generated by the secret key being kept private by the signer and by the contents of the message being signed. A signature has to be verifiable without using the signer's private key. Signatures should not be able to forge under chosen-message attacks. The ECDSA is generally used and is one of the important signature schemes today. The ECDSA which was first proposed in 1992 by Vanstone [25] is the elliptic curve analog of the DSA.

If Alice wants to send a digitally signed message to Bob, then at first, they choose a finite field F_q , an elliptic curve E , defined over that field and a generator point G (with order n). Alice's key pair is (d, Q) , where d is her private key and Q is her public key.



Protocol :

ECDSA Key Generation
1. Select a random integer $d \in [2, n-2]$.
2. Compute $Q = d * G$.
3. The public key of the user A is (E, G, n, Q) and the private key is d .

Table 4.2 ECDSA key generation.

ECDSA Signature Generation
1. Select a random integer $k \in [2, n-2]$.
2. Compute $kG = (x_1, y_1)$ and $r = x_1 \bmod n$.

<p>If then go to step 1.</p> <ol style="list-style-type: none"> 3. Compute $k^{-1} \bmod n$. 4. Compute $e = \text{SHA-1}(m)$. 5. Compute $s = k^{-1}(e + d \cdot r) \bmod n$. If $s = 0$ then go to step 1. Here SHA-1 is the secure hash algorithm of FIPS PUB 180-1. 6. Alice's signature for the message m is the pair of integer (r, s).
--

Table 4.3 ECDSA signature generation.

When Bob verifies Alice's signature (r, s) on the message m , he obtains an authentic copy of Alice's parameters and public key. Bob then does the following :

ECDSA Signature Verification
<ol style="list-style-type: none"> 1. Compute $e = \text{SHA-1}(M)$. 2. Compute $w = s^{-1} \bmod n$. 3. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$. 4. Compute $X = u_1G + u_2Q$. If $X = O$ then reject the signature. 5. Otherwise compute $v = x_1 \bmod n$ where $X = (x_1, y_1)$. 6. Accept the signature if and only if $v = r$.

Table 4.4 ECDSA signature verification.

From Table 4.3 we know that $s = k^{-1}(e + d \cdot r) \bmod n$, so we can derive Equation (4-1) as follows :

$$k \equiv s^{-1}(e + d \cdot r) \equiv s^{-1}e + s^{-1}rd \equiv we + wrd \equiv u_1 + u_2d \bmod n \quad (4-1)$$

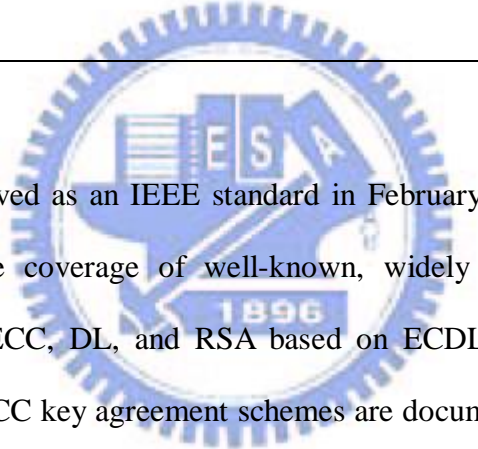
From Table 4.2 we see $Q = d * G$, thus $u_1G + u_2Q = (u_1 + u_2d)G = kG$ and so $v = r$ as require.

If the signature pair (r, s) on the message m was really generated by Alice, then Bob can accept the signature if and only if $v = r$.

4.4 Standards of Elliptic Curve Cryptography

There several organizations which develop standards such as International Standards Organization (ISO), Institute of Electrical and Electronics Engineers (IEEE) and Federal Information Processing Standards (FIPS). The organizations mentioned above are the most important for security in information technology.

IEEE P1363 [14]

The IEEE logo is a circular emblem with a gear-like border. Inside the circle, there is a shield with the letters 'E', 'S', and 'A' stacked vertically. Below the shield, the year '1896' is written. The shield is flanked by two stylized figures holding a banner.

IEEE P1363 was approved as an IEEE standard in February 2000. This document includes comprehensive coverage of well-known, widely marketed public key cryptography such as ECC, DL, and RSA based on ECDLP, DLP, and IF. ECC digital signatures and ECC key agreement schemes are documented in P1363. (ECC encryption and ECC key transport schemes will be specified in P1363a.) The latest draft is available at <http://grouper.ieee.org/groups/1363/index.html>.

FIPS (Federal Information Processing Standard) 186-2 : [12]

National Institute of Standards and Technology (NIST) announced FIPS 186-2 in February 2000. The standard is the extension of its Digital Signature Standard (DSS) and the ECDSA specified in ANSI X9.62 is included.

ANSI X9F

ECC is being incorporated into two American National Standards Institute (ANSI) Accredited Standards Committee (ASC) X9F (Financial Services) drafts.

- I ANSI X9.62 [17] – “Elliptic Curve Digital Signature Algorithm (ECDSA).”

ECDSA is the elliptic curve analogue of the Digital Signature Algorithm (DSA). The standard was published as an ANSI standard in January 1999.

- I ANSI X9.63 – “elliptic Curve Key Agreement and Key Management.”

X9.62 and X9.63 are both used to apply in the financial services industry.

ISO/IEC

ECC is being incorporated into several ISO/IEC drafts such as ISO/IEC 14888, ISO/IEC 9796-4 and ISO/IEC 14946.

- I ISO/IEC 14888 specifies digital signature with certificate-based mechanisms and provides an overview of various digital signature mechanisms.
- I ISO/IEC 9796-4 specifies digital signature with message recovery and discrete logarithm-based mechanisms.
- I ISO/IEC 14946 specifies cryptographic techniques based on elliptic curves.

Table 4.5 Standards of ECC.

Chapter 5

Implementation of Arithmetic Processor for ECC and Simulation Results

We introduce the original architecture for $GF(p)$ in Section 5.1. In Section 5.2, we present our modified architecture for dual-field $GF(p)$ and $GF(2^m)$. Finally, we show our simulation results in Section 5.3.

5.1 Architecture

We adopt the architecture which is introduced in [30], and will modify it from $GF(p)$ to support dual-field $GF(p)$ and $GF(2^m)$.

Their Elliptic Curve processor (ECP) can be divided into 5 levels hierarchically as shown in Figure 5.1.

The operation blocks on each level from top to bottom are as follows :

- I Level 1 : Main Controller (MC)
- I Level 2 :
 1. Affine to projective coordinates converter (A to P)
 2. Normal to Montgomery representation converter (N to M)
 3. EC point multiplier (EPM)
 4. Projective to affine coordinates converter (P to A)

5. Montgomery to normal representation converter (M to N)

I Level 3 :

1. EC Point doubling, addition circuit (EPDA)
2. Modular Multiplicative Inverter (MMI)

I Level 4 :

1. Montgomery Modular Multiplication Circuit (MMMC)
2. Modular Addition, Subtraction circuit (MASC)

I Level5 : Addition, Subtraction circuit (ASC)

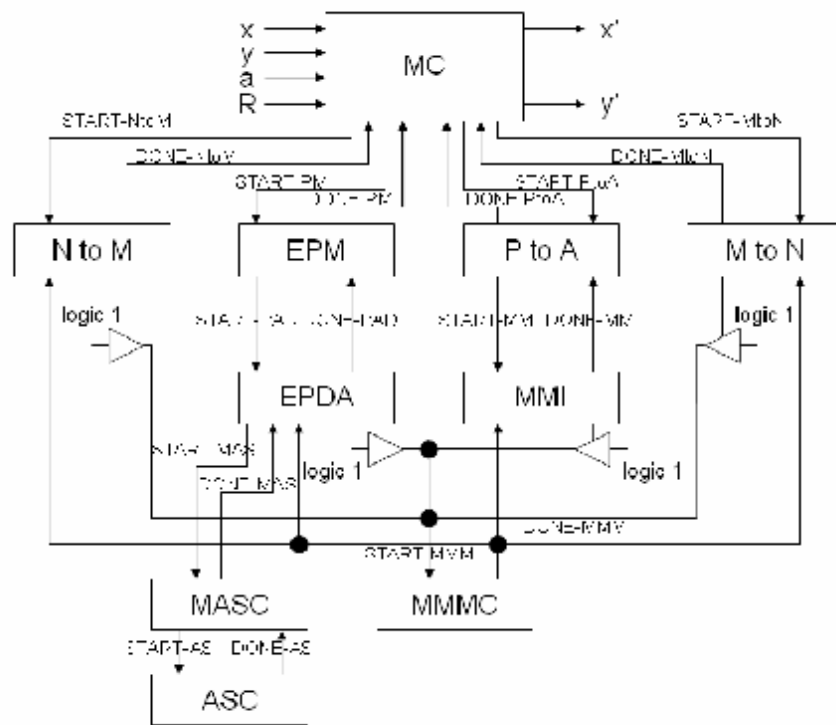


Figure 5.1 EC point multiplier circuit block diagram.

5.1.1 MC, NtoM, EPM, PtoA and MtoN

I Main Controller (MC)

The START signal is the instruction signal from host. MC instructs, NtoM to start conversion from normal to Montgomery representation, EPM to start point

multiplication, PtoA to start conversion from projective to affine coordinates and MtoN to start a conversion from Montgomery representation one after another by setting START-NtoM, START-PM, START-PtoA and START-MtoN signals, respectively. The DONE-NtoM, DONE-PM, DONE-PtoA and DONE-MtoN signals indicate that the related operations are finished. The DONE signal indicates to the host that a complete point multiplication operation is finished and the results are ready on output ports.

I Normal to Montgomery representation converter (NtoM)

The conversion of an integer x from the normal representation to the Montgomery representation is done as $MM(x, R^2) = xR^2R^{-1} \bmod M$. Multiplication by MMMC of two numbers that are in Montgomery representation will produce the Montgomery representation of product as $MM(xR, yR) = xRyRR^{-1} \bmod M = xyR \bmod M$. Modular addition and subtraction of two numbers that are in Montgomery representation will produce the Montgomery representation of the sum or difference as $xR \bmod M \pm yR \bmod M = (x \pm y)R \bmod M$. Because of these relations; the Montgomery representation of the coordinates of P , the coefficient a and number l will be calculated in the beginning of point multiplication by the NtoM circuit and all the operations during the EC point multiplication will be done in Montgomery representation.

NtoM makes MMMC to execute 4 MMMs, $MM(1, R^2) = R \bmod M$,
 $MM(x, R^2) = xR \bmod M$, $MM(y, R^2) = yR \bmod M$, $MM(a, R^2) = aR \bmod M$.

I EC Point Multiplier (EPM)

EPM controls the execution of the Elliptic Curve Point Multiplication Algorithm. The circuit stays in IDLE state until the START-PM signal from the MC is set.

DONE-PM signal indicates that the scanning of the bits of k is finished, so the result of the operation can be read from the output ports. EPM instructs EPDA to start a point double operation by setting START-PAD signal and resetting ADD-DOUBLE signal and a point addition operation by setting START-PAD and ADD-DOUBLE signals. DONE-PAD from EPDA indicates the a point double or addition operation is finished.

Elliptic Curve Point Multiplication Algorithm is shown below :

Algorithm 1 : Elliptic Curve Point Multiplication

Input: EC point $P = (x, y)$, integer $k = (k_{n-1}, \dots, k_0)_2$

Output: $Q = (x', y')$

1. $Q \leftarrow P$
2. for i from $n - 2$ downto 0 do
3. $Q \leftarrow 2Q$
4. if $k_i = 1$ then
5. $Q \leftarrow Q + P$
6. end if
7. end for

I Projective to affine coordinates converter (PtoA)

After finishing the EC point multiplication the result point Q must be converted from J^m coordinates to affine coordinates. This is done as $(X, Y, Z, aZ^4) \rightarrow (x, y)$ such that $x = XZ^{-2}$ and $y = YZ^{-3}$ [31].

PtoA waits in IDLE state until the signal START-PtoA from MC is set. After it is set, PtoA visits the other five states in the following order and after DONE-MMM signal from MMM circuit is set in (PtoA-S5) state, PtoA sets DONE-PtoA signal and goes back to IDLE state.

1. PtoA-S1 : $Z^1R = \text{Modular Multiplicative Inversion of } Z$
2. PtoA-S2 : $Z^2R = MM(Z^1R, Z^1R)$

3. PtoA-S3 : $xR = XZ^2R = MM(XR, Z^2R)$
4. PtoA-S4 : $Z^3R = MM(Z^1R, Z^2R)$
5. PtoA-S5 : $yR = YZ^3R = MM(YR, Z^3R)$

I Montgomery to Normal representation converter (MtoN)

Because the coordinates of the product point must be in normal representation, as a last action a conversion from Montgomery representation to normal representation is needed. This conversion requires two additional execution of the MMM operation with the inputs xR and 1, then yR and 1, as $x = MM(xR, 1) = xRR^{-1}$, $y = MM(yR, 1) = yRR^{-1}$.

5.1.2 EPDA , MMI and MASC

I EC Point doubling, addition (EPDA)

When converting the input point P from affine coordinates to projective coordinates we take Z as 1. The J^m representation of $P(x, y)$ is $(x, y, 1, a)$. During the execution of point multiplication one of the points to be added is always P .

Algorithm 2 : EC point addition and doubling

Input : $P_1 = (x, y, 1, a), P_2 = (X_2, Y_2, Z_2, aZ_2^4)$	Input : $P_1 = (X_1, Y_1, Z_1, aZ_1^4)$
Output : $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3, aZ_3^4)$	Output : $P_3 = 2P_1 = (X_3, Y_3, Z_3, aZ_3^4)$
<ol style="list-style-type: none"> 1. $T_1 \leftarrow Z_2^2$ 2. $T_2 \leftarrow xT_1$ 3. $T_1 \leftarrow T_1Z_2$ $T_3 \leftarrow X_2 - T_2$ 4. $T_1 \leftarrow yT_1$ 5. $T_4 \leftarrow T_3^2$ $T_5 \leftarrow Y_2 - T_1$ 6. $T_2 \leftarrow T_2T_4$ 7. $T_4 \leftarrow T_4T_3$ $T_6 \leftarrow 2T_2$ (a) 8. $Z_3 \leftarrow Z_2T_3$ $T_6 \leftarrow T_4 + T_6$ 9. $T_3 \leftarrow T_3^2$ 10. $T_1 \leftarrow T_1T_4$ $X_3 \leftarrow T_3 - T_6$ 11. $aZ_3^4 \leftarrow Z_3^2$ $T_2 \leftarrow T_2 - X_3$ 12. $T_3 \leftarrow T_3T_2$ 13. $aZ_3^4 \leftarrow (aZ_3^4)^2$ $Y_3 \leftarrow T_3 - T_1$ 14. $aZ_3^4 \leftarrow a(aZ_3^4)$ 	<ol style="list-style-type: none"> 1. $T_1 \leftarrow Y_1^2$ $T_2 \leftarrow 2X_1$ 2. $T_3 \leftarrow T_1^2$ $T_2 \leftarrow 2T_2$ 3. $T_1 \leftarrow T_2T_1$ $T_3 \leftarrow 2T_3$ 4. $T_2 \leftarrow X_1^2$ $T_3 \leftarrow 2T_3$ 5. $T_4 \leftarrow Y_1Z_1$ $T_3 \leftarrow 2T_3$ 6. $T_5 \leftarrow T_3(aZ_1^4)$ $T_6 \leftarrow 2T_2$ 7. $T_2 \leftarrow T_6 + T_2$ (b) 8. $T_2 \leftarrow T_2 + (aZ_1^4)$ 9. $T_6 \leftarrow T_2^2$ $Z_3 \leftarrow T_4$ 10. $T_4 \leftarrow 2T_1$ 11. $X_3 \leftarrow T_6 - T_4$ 12. $T_1 \leftarrow T_1 - X_3$ 13. $T_2 \leftarrow T_2T_1$ $aZ_3^4 \leftarrow 2T_5$ 14. $Y_3 \leftarrow T_2 - T_3$

According to these properties we can take $Z_1 = 1$ for EC point addition. Because there are both MMMC and MAS circuits available, these operations can be executed in parallel. EC point addition and doubling can be realized by Algorithm 2 (a) and (b), respectively.

I Modular Multiplicative Inverter (MMI)

Modular multiplicative inversion is done according to Fermat's theorem which is introduced in Section 2.2, $a^{-1} \equiv a^{p-2} \pmod{p}$, if $\gcd(d, p) = 1$. Because the order n of the generator point used in the ECDSA operation is limited to a prime number, a multiplicative inversion is executed by using Equation (2.4).

I Modular Addition, Subtraction Circuit (MASC)

Modular addition and subtraction are executed according to Algorithm 3 [21].

Algorithm 3 : Modular addition and subtraction

Input : $M, 0 \leq A < M, 0 \leq B < M$

Input : $M, 0 \leq A < M, 0 \leq B < M$

Output : $C = A + B \pmod{M}$

Output : $C = A - B \pmod{M}$

1. $C' = A + B$

1. $C' = A - B$

2. $C'' = C' - M$

2. $C'' = C' + M$

3. if $C'' < 0$ then

3. if $C' < 0$ then

4. $C = C'$

4. $C = C''$

5. else

5. else

6. $C = C''$

6. $C = C'$

7. end if

7. end if

5.1.3 MMMC

Because we want to propose a scalable Montgomery modular multiplication circuit, we replace the MMMC in [30] with the scalable MMMC in [32]. In this section, we will introduce the scalable MMMC [32] roughly.

The original Montgomery Algorithm is shown below :

Algorithm 4 : The original Radix-2 Montgomery multiplication

Input : m-bit operands $X = (x_{m-1}, \dots, x_1, x_0), Y$ and $M, 0 \leq X, Y < M$

Output : S_m

initial $S = 0$

for $i = 0$ to $m - 1$

 if $(S_i + x_i Y)$ is even

 then $S_{i+1} = (S_i + x_i Y) / 2$

 else $S_{i+1} = (S_i + x_i Y + M) / 2$

if $S_m \geq M$ then $S_m = S_m - M$

They propose an algorithm in which the operand Y (multiplicand) is scanned word-by-word, and the operand X (multiplier) is scanned bit-by-bit. This decision enables them to obtain an efficient hardware implementation. They call it *Multiple Word Radix-2 Montgomery Multiplication algorithm (MWR2MM)*. They make use of the following vectors :

$$M = (M^{(e-1)}, \dots, M^{(1)}, M^{(0)}),$$

$$S = (S^{(e-1)}, \dots, S^{(1)}, S^{(0)}),$$

$$Y = (Y^{(e-1)}, \dots, Y^{(1)}, Y^{(0)}),$$

$$X = (x_{m-1}, \dots, x_1, x_0),$$

where the words are marked with superscripts and the bits are marked with subscripts. The concatenation of vectors a and b is represented as $(a; b)$. A particular range of bits in a vector a from position i to position $j, j > i$ is represented as $a_{j \dots i}$. The bit position i of the k^{th} word of a is represented as $a_i^{(k)}$. The details of the *MWR2MM* algorithm are given below :

Algorithm 5 : The MWR2MM

1. $S = 0$
2. for $i = 0$ to $m - 1$
3. $(C, S^{(0)}) = x_i Y^{(0)} + S^{(0)}$
4. if $S_0^{(0)} = 1$ then
5. $(C, S^{(0)}) = (C, S^{(0)}) + M^{(0)}$
6. for $j = 1$ to $e - 1$
7. $(C, S^{(j)}) = C + S^{(j)} + x_i Y^{(j)} + M^{(j)}$
8. $S^{(j-1)} = (S_0^{(j)}, S_{w-1, \dots, 1}^{(j-1)})$
9. $S^{(e-1)} = (C, S_{w-1, \dots, 1}^{(e-1)})$
10. else
11. for $j = 1$ to $e - 1$
12. $(C, S^{(j)}) = C + S^{(j)} + x_i Y^{(j)}$
13. $S^{(j-1)} = (S_0^{(j)}, S_{w-1, \dots, 1}^{(j-1)})$
14. $S^{(e-1)} = (C, S_{w-1, \dots, 1}^{(e-1)})$
15. If $S \geq M$ then $S = S - M$

The dependency graph for the *MWR2MM* algorithm is shown in Figure 5.2. Each circle in the graph represents an atomic computation and is labeled according to the type of action performed. Task A corresponds to three steps : (1) test the least significant bit of S to determine if M should be added to S during this and next steps, (2) addition of words from S , $x_i Y$, and M (depending on the test performed), and (3) one-bit right shift of a S word. Task B corresponds to steps (2) and (3). We observe from this graph that the degree of parallelism and pipelining can be very high. Each column in the graph may be computed by a separate processing element (PE), and the data generated from one PE may be passed to another PE in a pipelined fashion.

A pipelined organization for the system is shown in Figure 5.3. The pipeline itself was named kernel in the figure and it is composed of p PEs. The other blocks represent memory, data conversion, and control unit. Each processing element in the

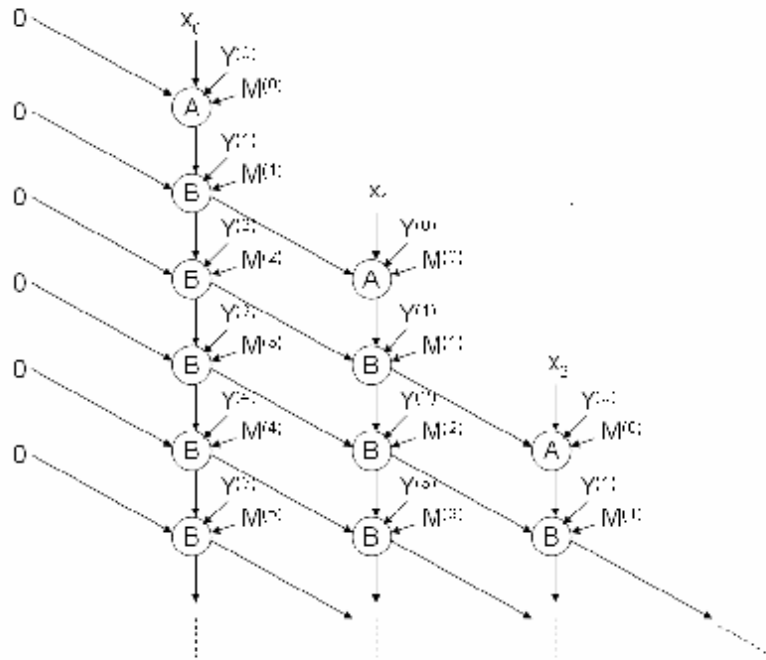


Figure 5.2 The dependency graph for the *MWR2MM* Algorithm.

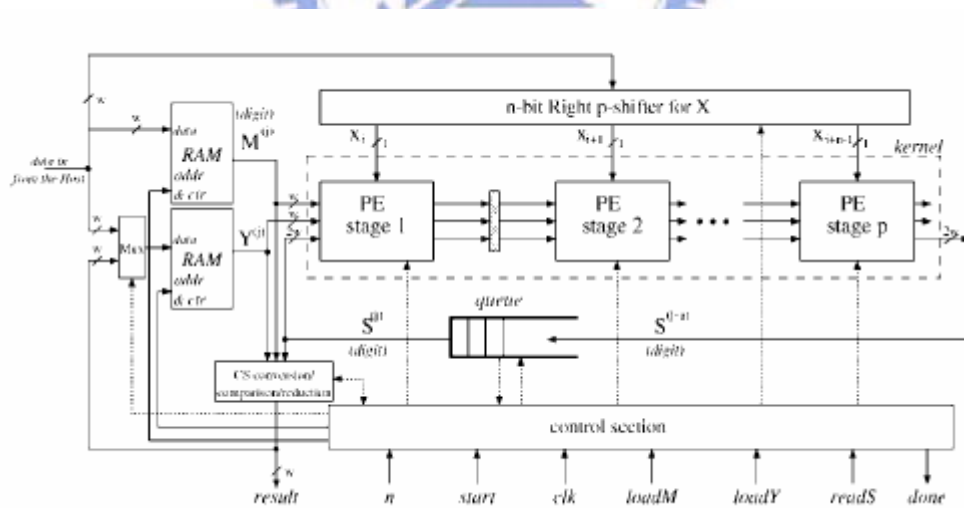


Figure 5.3 Pipelined organization for the multiplier.

pipeline relays the received words to the next downstream unit. All paths are w -bits wide, except for the x_i inputs (only 1 bit). The kernel itself does not limit the final computation precision. If more precision is required, it is only necessary to feed more

words. The final and intermediate results are stored in the queue. Gray boxes indicate registers.

The control block function can be inferred from the algorithm description that was provided, combined with other data manipulation tasks that must be done to transfer data between the multiplier and the host system.

The data path design for the case $w = 3$ is shown in Figure 5.4(b). It has a more complicated shift and alignment section to generate the next S word. When computing the bits of word j (step j), the circuit generates $w-1$ bits of $S^{(j)}$, and the most significant bit of $S^{(j-1)}$. The bits of $S^{(j-1)}$ computed at step $j-1$ must be delayed and concatenated with the most significant bit generated at step j (alignment).

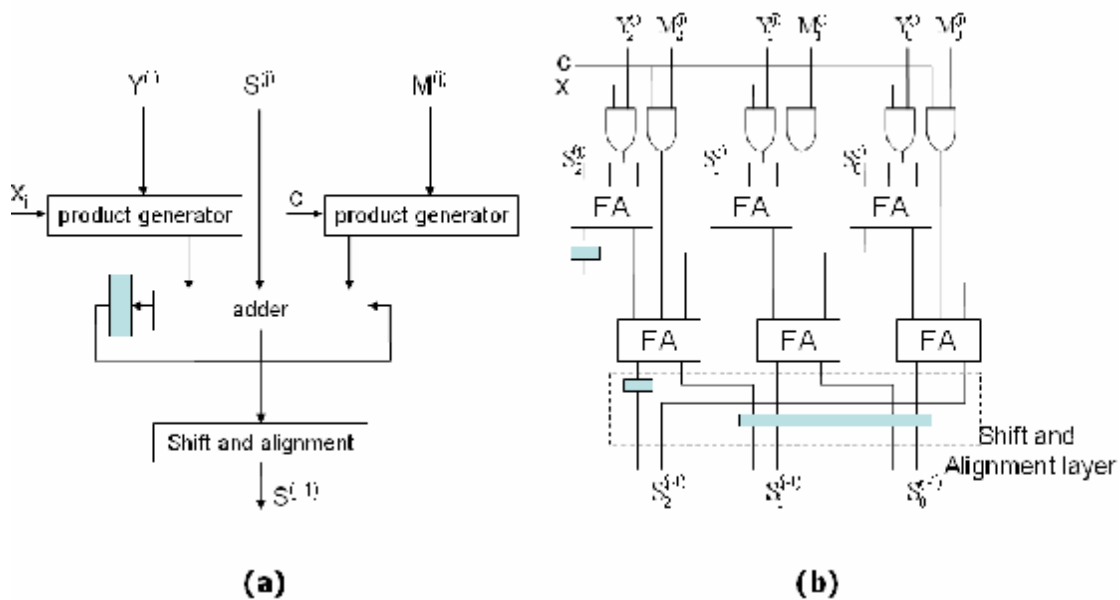


Figure 5.4 PE data path (a) block diagram and (b) logic diagram for $w = 3$ bits.

Finally, Figure 5.5 illustrates what happens in last stage of the pipeline. A pair of redundant words $(TC_j^{(i)}, TS_j^{(i)})$ are generated each cycle for e clock cycles. The word adder can be used to add these pairs in order to obtain the result words $C^{(i)}$. Note that

only one extra cycle is needed to convert the result from the Carry-Save form to the nonredundant form.

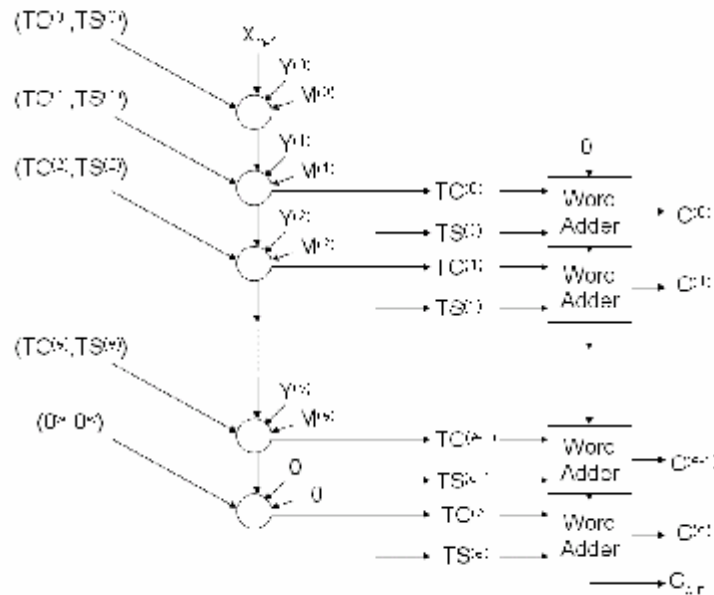


Figure 5.5 Converting the result from the Carry-Save form to the nonredundant form in the last stage of the pipeline.

An example of the computation for 7-bit operands is shown in Figure 5.6 for the word size $w = 1$ provided that there are sufficient numbers of PEs preventing the pipeline to stall. Note that there is a delay of 2 clock cycles between the stage for x_i and the stage for x_{i+1} . The total execution time for the computation takes 20 clock cycles in this example.

If there are at least $\lceil e+1/2 \rceil$ PEs in the pipeline organization the pipeline stalls do not take place. For the example in Figure 5.7 ,less than $\lceil 7+1/2 \rceil = 4$ PEs cause the pipeline to stall. Figure 5.7 shows what happens if there are only three PEs available for the same example.

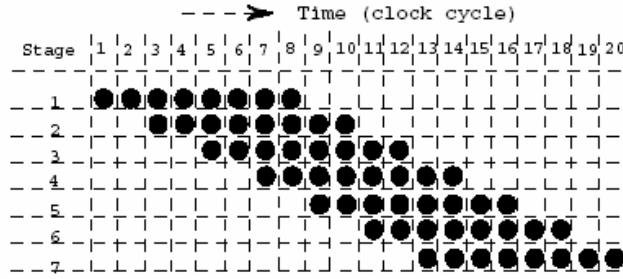


Figure 5.6 An example of pipeline computation for 7-bit operands, where $w = 1$.

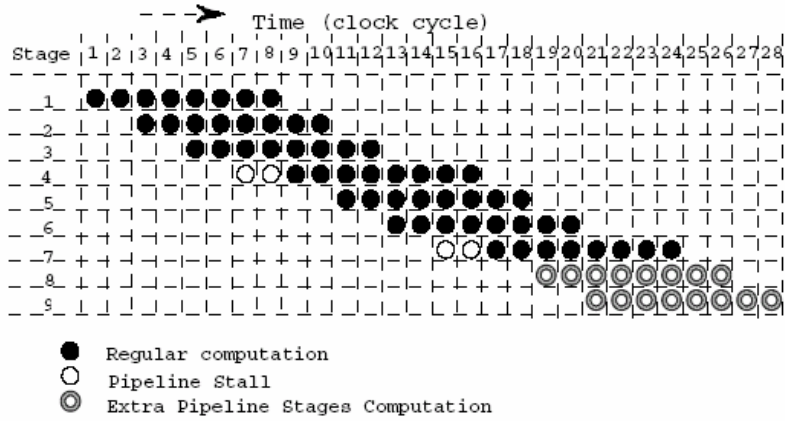


Figure 5.7 An example of pipeline computation for 7-bit operands, illustrating the situation of pipeline stalls, where $w = 1$.

5.2 Modified Architecture for Dual-field

We modify the architecture introduced in Section 5.1. Thus, the architecture can support dual-field $GF(p)$ and $GF(2^m)$.

5.2.1 Modified NtoM, EPDA, PtoA and MMI

I Normal to Montgomery (NtoM)

NtoM makes MMMC to execute 3 MMMs, $MM(1, R^2) = R \bmod M$, $MM(x, R^2) = xR \bmod M$, $MM(y, R^2) = yR \bmod M$.

I EC Point doubling, addition (EPDA)

Because we want to make our architecture to support dual-field, the select of projective coordination is very important. From the comparison in Table 3.2, we choose the projective coordinate $(X/Z, Y/Z^2)$. That is,

For $GF(p)$

Let $P_1 = (X_1/Z_1, Y_1/Z_1^2)$ and $P_2 = (X_2/Z_2, Y_2/Z_2^2)$ be two points on the elliptic curve E , then the addition formula is :

If $P_1 \neq P_2, Z_1 = 1$ (addition)

$$A = X_1Z_2, B = Z_2^2, D = X_2 + A, E = X_2 - A,$$

$$F = Y_2 - Y_1B, X_3 = F^2 - Z_2DE^2, Z_3 = BE^2,$$

$$Y_3 = FZ_2E(X_1Z_3 - X_3) - Y_1Z_3^2$$

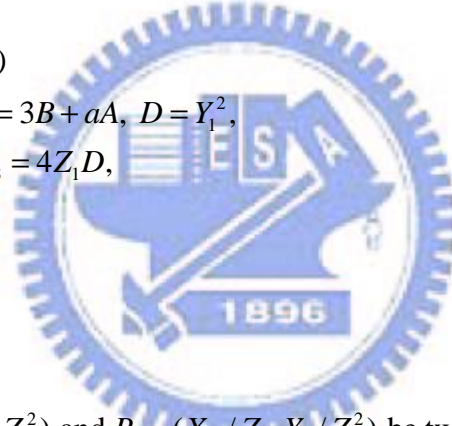
If $P_1 = P_2$ (doubling)

$$A = Z_1^2, B = X_1^2, C = 3B + aA, D = Y_1^2,$$

$$E = X_1Z_3 - X_3Z_1, Z_3 = 4Z_1D,$$

$$X_3 = Z_1C^2 - 8X_1D,$$

$$Y_3 = 2Y_1CE - 16D^2Y_1$$



For $GF(2^m)$

Let $P_1 = (X_1/Z_1, Y_1/Z_1^2)$ and $P_2 = (X_2/Z_2, Y_2/Z_2^2)$ be two points on the elliptic curve E , then the addition formula is :

If $P_1 \neq P_2, Z_1 = 1$ (addition)

$$U = Z_2^2Y_1 + Y_2, S = Z_2X_1 + X_2, T = Z_2S$$

$$Z_3 = T^2, V = Z_3X_1, C = (X_1 + Y_1)$$

$$X_3 = U^2 + T(U + S^2 + Ta_2)$$

$$Y_3 = (V + X_3)(TU + Z_3) + Z_3^2C$$

If $P_1 = P_2$ (doubling)

$$Z_3 = Z_1^2X_1^2$$

$$X_3 = X_1^4 + a_6Z_1^4$$

$$Y_3 = a_6Z_1^4Z_3 + X_3(a_2Z_3 + Y_1^2 + a_6Z_1^4)$$

Therefore, in $GF(p)$, addition operation needs 14M and doubling operation needs 14M. In $GF(2^m)$, addition operation needs 9M+5S and doubling operation needs 5M+5S. Similar to Algorithm 2, in EPDA, we can get modified Algorithm 6 and 7 shown below :

Algorithm 6 : EC point addition and doubling in $GF(p)$

Input : $P_1 = (x, y, 1), P_2 = (X_2, Y_2, Z_2)$	Input : $P_1 = (X_1, Y_1, Z_1)$
Output : $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$	Output : $P_3 = 2P_1 = (X_3, Y_3, Z_3)$
<ol style="list-style-type: none"> 1. $T_1 \leftarrow X_1 Z_2$ 2. $T_2 \leftarrow Z_2^2$ 3. $T_5 \leftarrow Y_1 T_2$ 4. $T_3 \leftarrow Z_2 T_3$ 5. $T_1 \leftarrow T_4^2$ 6. $Z_3 \leftarrow T_2 T_1$ 7. $T_2 \leftarrow T_1 T_3$ 8. $T_3 \leftarrow T_5^2$ 9. $T_1 \leftarrow Z_2 T_5$ 10. $T_2 \leftarrow T_1 T_4$ 11. $T_3 \leftarrow X_1 Z_3$ 12. $T_4 \leftarrow Z_3^2$ 13. $T_1 \leftarrow T_2 T_5$ 14. $T_2 \leftarrow Y_1 T_4$ 15. $Y_3 \leftarrow T_1 - T_2$ 	<ol style="list-style-type: none"> 1. $T_1 \leftarrow Y_1^2$ 2. $T_2 \leftarrow Z_1^2$ 3. $T_3 \leftarrow X_1^2$ 4. $T_2 \leftarrow X_1^2$ 5. $Z_3 \leftarrow Z_1 T_1$ 6. $T_4 \leftarrow X_1 Z_3$ 7. $T_3 \leftarrow T_2^2$ 8. $T_3 \leftarrow Z_1 T_3$ 9. $T_5 \leftarrow X_1 T_5$ 10. $T_1 \leftarrow T_1^2$ 11. $T_5 \leftarrow X_3 Z_1$ 12. $T_1 \leftarrow Y_1 T_1$ 13. $T_2 \leftarrow T_2 T_4$ 14. $T_4 \leftarrow T_2 T_3$ 15. $Y_3 \leftarrow T_4 - T_1$
<ol style="list-style-type: none"> 2. $T_2 \leftarrow Z_2^2$ 3. $T_3 \leftarrow X_2 + T_1$ 4. $T_4 \leftarrow X_2 - T_1$ 5. $T_5 \leftarrow Y_2 T_5$ 6. $T_3 \leftarrow X_2 + T_1$ 7. $T_4 \leftarrow X_2 - T_1$ 8. $T_5 \leftarrow Y_2 T_5$ 9. $X_3 \leftarrow T_3 - T_2$ 10. $T_2 \leftarrow T_2 - X_3$ 11. $T_2 \leftarrow T_2 - X_3$ 	<ol style="list-style-type: none"> 1. $T_1 \leftarrow 2T_1$ 2. $T_1 \leftarrow 2T_1$ 3. $T_4 \leftarrow 2T_3$ 4. $T_3 \leftarrow T_3 + T_4$ 5. $T_2 \leftarrow T_2 + T_3$ 6. $T_5 \leftarrow 2T_1$ 7. $X_3 \leftarrow T_3 - T_5$ 8. $T_2 \leftarrow 2T_2$ 9. $T_4 \leftarrow T_4 - T_5$ 10. $T_3 \leftarrow 2T_1$

Algorithm 7 : EC point addition and doubling in $GF(2^m)$

Input : $P_1 = (x, y, 1), P_2 = (X_2, Y_2, Z_2)$	Input : $P_1 = (X_1, Y_1, Z_1)$
Output : $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$	Output : $P_3 = 2P_1 = (X_3, Y_3, Z_3)$
<ol style="list-style-type: none"> 1. $T_1 \leftarrow Z_2^2$ 2. $T_2 \leftarrow X_1 Z_2$ 3. $T_3 \leftarrow Y_1 T_1$ 4. $T_4 \leftarrow T_2^2$ 5. $T_5 \leftarrow Z_2 T_2$ 6. $T_3 \leftarrow a_2 T_5$ 7. $T_3 \leftarrow T_1^2$ 8. $T_4 \leftarrow T_4 T_5$ 9. $Z_3 \leftarrow T_5^2$ 10. $T_5 \leftarrow T_1 T_5$ 11. $T_2 \leftarrow X_1 Z_3$ 12. $T_3 \leftarrow Z_3^2$ 13. $T_4 \leftarrow T_1 T_2$ 14. $T_3 \leftarrow T_3 T_5$ 15. $Y_3 \leftarrow T_3 + T_4$ 	<ol style="list-style-type: none"> 1. $T_1 \leftarrow Z_1^2$ 2. $T_2 \leftarrow X_1^2$ 3. $Z_3 \leftarrow T_1 T_2$ 4. $T_3 \leftarrow T_2^2$ 5. $T_4 \leftarrow T_1^2$ 6. $T_5 \leftarrow a_6 T_4$ 7. $T_1 \leftarrow Y_1^2$ 8. $T_2 \leftarrow a_2 Z_3$ 9. $T_1 \leftarrow Z_3 T_5$ 10. $T_2 \leftarrow X_3 T_3$ 11. $Y_3 \leftarrow T_1 + T_2$
<ol style="list-style-type: none"> 2. $T_2 \leftarrow X_2 + T_2$ 3. $T_1 \leftarrow Y_2 + T_3$ 4. $T_4 \leftarrow T_1 + T_4$ 5. $T_4 \leftarrow T_4 + T_5$ 6. $X_3 \leftarrow T_3 + T_4$ 7. $T_1 \leftarrow T_5 + Z_3$ 8. $T_2 \leftarrow T_2 + X_3$ 9. $T_5 \leftarrow X_1 + Y_1$ 	<ol style="list-style-type: none"> 3. $X_3 \leftarrow T_3 + T_5$ 4. $T_3 \leftarrow T_1 + T_5$ 5. $T_3 \leftarrow T_2 + T_3$

I Projective to affine coordinates converter (PtoA)

After finishing the EC point multiplication the result point Q must be converted from projective coordinates to affine coordinates. This is done as $(X, Y, Z) \rightarrow (x, y)$ such that $x = XZ^{-1}$ and $y = YZ^{-2}$.

PtoA waits in IDLE state until the signal START-PtoA from MC is set. After it is set, PtoA visits the other five states in the following order and after DONE-MMM signal from MMM circuit is set in (PtoA-S5) state, PtoA sets DONE-PtoA signal and goes back to IDLE state.

1. PtoA-S1 : $Z^{-1}R = \text{Modular Multiplicative Inversion of } Z$
2. PtoA-S2 : $xR = XZ^{-1}R = MM(XR, Z^{-1}R)$
3. PtoA-S3 : $Z^{-2}R = MM(Z^{-1}R, Z^{-1}R)$
4. PtoA-S4 : $yR = YZ^{-2}R = MM(YR, Z^{-2}R)$

I Modular Multiplicative Inverter (MMI)

Because the order n of the generator point used in the ECDSA operation is limited to a prime number, a multiplicative inversion is executed by using Equation (2.4) $a^{-1} \equiv a^{p-2} \pmod{p}$, if $\gcd(a, p) = 1$. So, multiplicative inversion can be done by modular exponentiation of a by $p-2$. By the way, because we only use one MMMC, we must adopt sequential modular exponentiation shown below :

Algorithm 8 : Sequential Modular Exponentiation

Input: $T, M, E=(e_{k-1}, \dots, e_1, e_0)$

Output: $T^E \pmod{M}$

```
initial R=1
1.if  $e_{k-1} = 1$  then  $R=T$ ;
2.for  $i = k - 2$  downto 0
3.  $R=R \times R \pmod{M}$ 
4. if  $e_i = 1$  then  $R=R \times T \pmod{M}$ ;
return R;
```

Using Fermat's theorem and sequential modular exponentiation, the inverse requires about $1.5 \log_2 p$ multiplications. That is, if the prime $p = (p_{N-1}, \dots, p_1, p_0)$, the inverse requires about $1.5N$ multiplications.

5.2.2 Modified MMMC

From [30], it can be easily proved that $MM(S, 1) \leq M$, if $0 \leq S < 2M$. We can rewrite Algorithm 4 as Algorithm 9 without final subtraction.

Algorithm 9 : Radix-2 Montgomery multiplication without final subtraction

Input : $X = (0, x_m, \dots, x_1, x_0)_2, Y = (0, y_m, \dots, y_1, y_0)_2, M = (m_{m-1}, \dots, m_1, m_0)_2$
with $X, Y \in [0, 2M - 1], \gcd(M, 2) = 1$

Output : $S_{m+2} = XY 2^{-(m+2)} \bmod 2M$

initial $S = 0$

for $i = 0$ to $m + 1$

 if $(S_i + x_i Y)$ is even

 then $S_{i+1} = (S_i + x_i Y) / 2$

 else $S_{i+1} = (S_i + x_i Y + M) / 2$

The $(m+1)$ -bits operands are split into w -bit words. For now, suppose that e words are used. Word and bit vectors are represented as $M = (0, M^{(e-1)}, \dots, M^{(1)}, M^{(0)})$, $Y = (0, Y^{(e-1)}, \dots, Y^{(1)}, Y^{(0)})$, $S = (0, S^{(e-1)}, \dots, S^{(1)}, S^{(0)})$, $X = (0, x_m, \dots, x_1, x_0)$. M , Y and S are extended to $e+1$ words by a mostsignificant zero word. Then, from Algorithm 9, we can derive the Algorithm 10 : MWR2MM for dual-field.

Algorithm 10 : The MWR2MM for Dual-Field

$GF(p)$	$GF(2^m)$
1. $S = 0, C = 0$	1. $S = 0$
2. for $i = 0$ to $m+1$	2. for $i = 0$ to $m-1$
3. $(C, S^{(0)}) = x_i Y^{(0)} + S^{(0)}$	3. $S^{(0)} = x_i Y^{(0)} + S^{(0)}$
4. if $S_0^{(0)} = 1$ then	4. if $S_0^{(0)} = 1$ then
5. $(C, S^{(0)}) = (C, S^{(0)}) + M^{(0)}$	5. $S^{(0)} = S^{(0)} + M^{(0)}$
6. for $j = 1$ to e	6. for $j = 1$ to $e-1$
7. $(C, S^{(j)}) = C + S^{(j)} + x_i Y^{(j)} + M^{(j)}$	7. $S^{(j)} = S^{(j)} + x_i Y^{(j)} + M^{(j)}$
8. $S^{(j-1)} = (S_0^{(j)}, S_{w-1, \dots, 1}^{(j-1)})$	8. $S^{(j-1)} = (S_0^{(j)}, S_{w-1, \dots, 1}^{(j-1)})$
9. $S^{(e)} = (C, S_{w-1, \dots, 1}^{(e)})$	9. $S^{(e-1)} = (0, S_{w-1, \dots, 1}^{(e-1)})$
10. else	10. else
11. for $j = 1$ to e	11. for $j = 1$ to $e-1$
12. $(C, S^{(j)}) = C + S^{(j)} + x_i Y^{(j)}$	12. $S^{(j)} = S^{(j)} + x_i Y^{(j)}$
13. $S^{(j-1)} = (S_0^{(j)}, S_{w-1, \dots, 1}^{(j-1)})$	13. $S^{(j-1)} = (S_0^{(j)}, S_{w-1, \dots, 1}^{(j-1)})$
14. $S^{(e)} = (C, S_{w-1, \dots, 1}^{(e)})$	14. $S^{(e-1)} = (0, S_{w-1, \dots, 1}^{(e-1)})$

Observing Algorithm 10, there is one difference between $GF(p)$ and $GF(2^m)$. That is, it needs to consider carry out over $GF(p)$ but it does not need over $GF(2^m)$. Therefore, we can replace Full adders (FA) in Figure 5.3 with our proposed Dual-field adder (DFA).

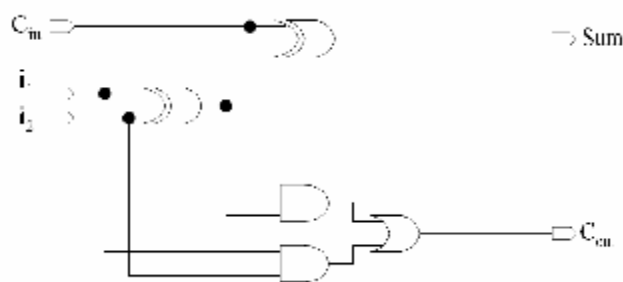


Figure 5.8 Standard full adder.

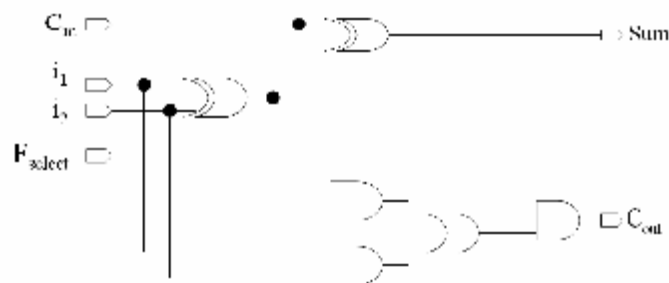


Figure 5.9 Dual-field adder.

5.3 Simulation Results

In this section, we will discuss the synthesis result of our design. We use TSMC 0.25 μ m process technology and Synopsys Design Analyzer to synthesize our RTL code. The gate count of our design is the physical cell size reported by Synopsys Design Analyzer divided by the size of two input NAND gate (it has 4 transistors), NAND2X1. Our architecture uses 8PEs with wordsize = 8bits to perform 192 bits input. According to implementation results, the gate count is 26,774 (divided 12,949 gates for the core and 13,825 gates for the memory) and Minimum clock period is 3.5ns (Maximum clock rate 285.7MHz). We show the comparison of ECC hardware performance in Table 5.1 and Table 5.2 for $GF(p)$ and $GF(2^m)$, respectively. In Table 5.3, we compare the circuit size of our designed with [33] and [30]. Finally, the latency of the operations according to the clock frequency of the implemented circuit is given in Table 5.4 for $GF(p)$ and Table 5.5 for $GF(2^m)$.

Reference	Field size (bits)	Platform	# of cycles	Max. freq. (MHz)	Operation time (ms)	Notes
Our work	192	0.25- μ m COMS ASIC	691	285.7	0.00242	8PEs with w=8bits
[33]	192	0.13- μ m COMS ASIC	1345	363.6	0.0037	8x8-bits multiplier
[30]	160	FPGA	484	91.308	0.0053	Systolic array

Table 5.1 $GF(p)$ ECC hardware performance comparison.

Reference	Field size (bits)	Platform	# of cycles	Max. freq. (MHz)	Operation time (ms)	Notes
Our work	192	0.25- μ m COMS ASIC	591	285.7	0.00207	8PEs with w=8bits
[33]	192	0.13- μ m COMS ASIC	1269	763.4	0.0017	8x8-bits multiplier

Table 5.2 GF(2^m) ECC hardware performance comparison.

Reference	Field size (bits)	Platform	Core size (gates)	Memory size (gates)	Total size (gates)	Notes
Our work	192	0.25- μ m COMS ASIC	12,949	13825	26,774	8PEs with w=8bits
[33]	192	0.13- μ m COMS ASIC	19,935	9,720	29,655	8x8-bits multiplier
[30]	160	FPGA			115,520	Systolic array

Table 5.3 Circuit size comparison.

Operations	Sub-operations	Execution time* <i>ms</i>
NtoM	3MMM	0.00726
EPM	n EC point doubling+ $n/2$ EC point addition	9.75188
PtoA	MMI+3MMM	0.70382
MtoN	2MMM	0.00484
EC point doubling	14 MMM	0.03386
EC point addition	14 MMM	0.03386
MMI	1.5N MMM	0.69656
MMM		0.00242

*For $N=n=192$ at 285.7 MHz.

Table 5.4 GF(p) latency of the operations executed in ECP.

Operations	Sub-operations	Execution time* <i>ms</i>
NtoM	3MMM	0.00621
EPM	n EC point doubling+ $n/2$ EC point addition	6.75192
PtoA	MMI+3MMM	0.60196
MtoN	2MMM	0.00414
EC point doubling	10 MMM	0.02069
EC point addition	14 MMM	0.02896
MMI	1.5N MMM	0.59576
MMM		0.00207

*For $N=n=192$ at 285.7 MHz.

Table 5.5 GF(2^m) latency of the operations executed in ECP.



Chapter 6

Conclusions

We have described an efficient and scalable implementation of an elliptic curve cryptosystem over dual-fields $GF(p)$ and $GF(2^m)$. The processor can be programmed to execute a modular multiplication, addition/subtraction, multiplicative inversion, EC point addition/doubling and multiplication. We use the method of Montgomery for modular multiplication. Besides, we can find that addition and doubling formula need inverse operations in affine coordinate system. As a field inversion is still far more expensive than a field multiplication, we use a method that changes the affine coordinate system to projective coordinate system. We only need field multiplications in projective coordinate system and finally need only one inversion operation when PtoA. By the coordinate system changing, we save much time to implement the ECC hardware. In other words, this speeds up the cryptosystem computation. Compare with others paper, our ECC design has relative smaller area and can provide better performance. Our design also provides scalability and can used for various applications.

Bibliography

- [1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 644-654, 1976.
- [2] T.ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, pp. 473-481, 1985.
- [3] R. Rivest, A. Shamir and L.M. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the ACM*, vol. 21, pp. 120-126, 1978.
- [4] Victor S. Miller, "Use of Elliptic Curves in Cryptography," *Advances in Cryptology – CRYPTO '85 Proceedings, Lecture Notes in Computer Science, (1986) Springer-Verlag*, Pg. 417-426.
- [5] Neal Koblitz, "Elliptic Curves Cryptosystems," *Mathematics of Computation*, 48n.177 (1987), Pg. 203-209.
- [6] G. Agnew, R. Mullin, I. Onyszchuk, and S. Vanstone, "An Implementation of Elliptic Curve Cryptosystems over F_2^{155} ," *IEEE J. Selected Areas Comm.*, vol. 11, pp. 804-813, June 1993.
- [7] S. Sutikno, A. Surya, and R. Effendi, "An Implementation of ElGamal Elliptic Curves Cryptosystems," *Proc. 1998 IEEE Asia-Pacific Conf. Circuits and Systems (APCCAS '98)*, pp. 483-486, Nov. 1998.
- [8] S. Sutikno, R. Effendi, and A. Surya, "Design and Implementation of Arithmetic Processor F_2^{155} for Elliptic Curve Cryptosystems," *Proc. 1998 IEEE Asia-Pacific Conf. Circuits and Systems (APCCAS '98)*, pp. 647-650, Nov. 1998.

- [9] K.H. Leung, K.W. Ma, W.K. Wong, and P.H.W. Leong, "FPGA Implementation of a Microcoded Elliptic Curve Cryptographic Processor," *Proc. 2000 IEEE Symp. Field Programmable Custom Computing Machines (FCCM '99)*, pp. 68-76, Apr. 2000.
- [10] M. Ernst, S. Klupsch, O. Hauck, and S.A. Huss, "Rapid Prototyping for Hardware Accelerated Elliptic Curve Public-Key Cryptosystems," *Proc. 12th Int'l Workshop Rapid System Prototyping (RSP 2001)*, pp. 24-29, June 2001.
- [11] L. Gao, S. Shrivastava, and G. Sobelman, "Elliptic Curve Scalar Multiplier Design Using FPGAs," *Proc. Cryptographic Hardware and Embedded Systems (CHES '99)*, pp. 257-268, Aug. 1999.
- [12] M.C. Rosner, "Elliptic Curve Cryptosystems on Reconfigurable Hardware," *master's thesis, Worcester Polytechnic Inst., May 1998*,
http://www.ece.wpi.edu/research/crypt/publications/documents/ms_mrosner.pdf
- [13] G. Orlando and C. Paar, "A High-Performance Reconfigurable Elliptic Curve Processor for $GF(2^m)$," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2000)*, pp. 41-56, Aug. 2000.
- [14] N.P. Smart, "The Hessian Form of an Elliptic Curve," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2001)*, pp. 118-125, May 2001.
- [15] S. Okada, N. Torii, K. Itoh, and M. Takenaka, "Implementation of Elliptic Curve Cryptographic Coprocessor over $GF(2^m)$ on an FPGA," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2000)*, pp. 25-40, Aug. 2000.
- [16] J. Goodman and A. Chandrakasan, "An Energy Efficient Reconfigurable Public-Key Cryptography Processor Architecture," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2000)*, pp. 175-190, Aug. 2000.
- [17] G. Orlando and C. Paar, "A Scalable $GF(p)$ Elliptic Curve Processor Architecture for Programmable Hardware," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2001)*, pp. 349-363, May 2001.

- [18] S. Xu and L. Batina, "Efficient Implementation of Elliptic Curve Cryptosystems on an ARM7 with Hardware Accelerator," *Proc. Information Security (ISC 2001)*, pp. 266-3279, Oct. 2001.
- [19] V.S. Miller, "Use of Elliptic Curve in Cryptography," *Proc. Advances in Cryptology (Crypto '85)*, pp. 417-426, 1986.
- [20] N. Koblitz, "Elliptic Curve Cryptosystems," *Math. Computing*, vol. 48, pp. 203-209, 1987.
- [21] N. Koblitz, "A Course in Number Theory and Cryptography," *volume 144 of Graduate text in mathematics.*, Springer-Verlag, Berlin, Germany, second edition, 1994.
- [22] "Digital Signature Standard (DSS)," *FIPS PUB 186-2, Nat'l Inst. of Standard Technology*, <http://csrc.nist.gov/publications/fips/fips186-2/fips186-2.pdf>, Jan. 2000.
- [23] IEEE P1363. Standard Specifications for Public-Key Cryptography, Draft. 13, IEEE Working Group, November 1999. <http://grouper.ieee.org/groups/1363/>
- [24] ANSI X9.62-1998, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), American Bankers Association, 1999.
- [25] S. Vanstone, "Responses to NIST's Proposal," *Communications of the ACM*, 35, July 1992, 50-52 (communicated by John Anderson).
- [26] A. Menezes, "ELLIPTIC CURVE PUBLIC KEY CRYPTOSYSTEMS," Kluwer Academic Publishers, Boston, 1993.
- [27] Essame Al-Daoud, Ramlan Mahmud, Mohammad Rushdan, and Adem Kiliçman, "A New Addition Formula for Elliptic Curve over $GF(2^n)$," *IEEE Transactions on Computers*, vol. 51, No. 8, AUGUST 2002.
- [28] J.S. Coron, H. Handschuh, and D. Naccache, "ECC : Do We Need to Count ?," *Advances in Cryptology-ASIACRYPT'99, LNCS 1716*, Springer, pp. 122-134, 1999.

- [29] Shu Lin and Daniel J. Costello, JR., “ERROR CONTROL CODING Fundamentals and Applications ”.
- [30] Siddika Berna Ors, Lejla Batina, Bart Preneel, Joos Vandewille, “Hardware Implementation of an Elliptic Curve Processor over $GF(p)$,” *Proc. Application-Specific Systems, Architectures, and Processors (ASAP 2003)*.
- [31] H. Cohen, A. Miyaji, and T. Ono, “Efficient elliptic curve exponentiation using mixed coordinates,” *Proc. of ASIACRYPT 1998, number 1514 in Lecture Notes in Computer Science*, pp. 51-65, Springer-Verlag, 1998.
- [32] Alexandre F. Tenca and Cetin K. Koc, “A Scalable Architecture for Montgomery Multiplication,” *Proc. First Int’l Workshop Cryptographic Hardware and Embedded Systems—CHES ’99, C. K. Koc, and C. Paar, eds.*, pp. 94-108, Aug. 1999.
- [33] Akashi Satoh and Kohji Takano, “A Scalable Dual-Field Elliptic Curve Cryptographic Processor,” *IEEE Transactions on Computers*, vol. 52, NO. 4, April 2003.

