4

Mathematical Background

The material presented in this Chapter, discusses several relevant mathematical concepts, fundamental for the understanding of elliptic curve public-key cryptosystems, the RSA algorithm, etc.. This material is also useful for a better understanding of the basic operations involved in the specifications of Rijndael algorithm (new Advanced Encryption Standard (AES)).

For a more detailed treatment of these aspects, the reader is referred to Number theory books like [376, 220, 47, 297], and to excellent cryptography books such as [226, 176, 129, 227, 106, 107]. The material presented in this chapter was written based on [56, 42, 289].

The rest of this Chapter is organized as follows. In Section 4.1 we give several basic definitions and theorems of the elementary theory of numbers. Then, in Section 4.2 we explain the concept of finite field, defining the associated arithmetic operations. Elliptic curves defined over $\mathbb{R}$ are described in Section 4.3. Thereafter, in Section 4.4, elliptic curves defined over binary extension fields are discussed in more detail. Several coordinate systems for representing elliptic curve points are presented in Section 4.5. Then different schemes for scalar representation are discussed in Section 4.6. Concluding remarks are given in Section 4.7.

4.1 Basic Concepts of the Elementary Theory of Numbers

Elementary theory of numbers is perhaps the single most important tool for developing cryptographic algorithms. Therefore, we start this chapter given some important definitions, theorems and results relevant to the subject of cryptography.

4.1.1 Basic Notions

Definition 4.1 (Integer Numbers). *Integer numbers are defined as the set of numbers* $\mathbb{Z} = \{..., -2, -1, 0, 1, 2, 3, ...\}$. *Within this set we have the subset of the natural numbers,* $\mathbb{N} = \{1, 2, 3, 4, ...\}$, *i.e., the subset of all positive numbers (greater than zero)*

Definition 4.2 (Divisibility). *Let* a *and* b *be two integers with* $a \neq 0$. *We say that* a *divides* b, *that* a *is a divisor or factor of* b, *that* b *is a multiple of* a *or that* b *is divisible by* a, *if there exists an integer* k *such that* $b = ak$. *This is written as* $a|b$. *If* a *does not divide* b *we write it as* $a \nmid b$.

Let a, b, $c \in \mathbb{Z}$, some important divisibility properties are,

i. For all $a \neq 0$, $a|a$. At the same time $1|b$ for all b,
ii. If $a|b$ then $a|bc$,
iii. If $a|b$ and $b|$ c then $a|c$,
iv. If $a|b$ and $a|c$ then $a|(b \pm c)$,
v. If $a|b$ and $a \nmid c$ then $a \nmid (b \pm c)$,
vi. If $a|b$ and $a|c$ then $a|(sb + tc)$ for any arbitrary integers s and t.

Theorem 4.3 (Integer division theorem). *Let* $a \in \mathbb{Z}$ *and* $b \in N$. *Then there exist* q, $r \in \mathbb{Z}$ *with* $0 \leq r < b$ *such that* $a = mq + r$. *Additionally,* q *and* r *are unique.*

Definition 4.4 (Greatest common divisor). *Given two integers* a *and* b *different than* 0, *we say that the integer* $d > 1$ *is the greatest common divisor, or gcd, of* a *and* b *if* $d|a$, $d|b$ *and for any other integer* c *such that* $c|a$ *and* $c|b$ *then* $c|d$. *In other words,* d *is the greatest positive number that divides both,* a *and* b.

Some of the properties of the greatest common divisor are,

- $gcd(a,b) = gcd(|a|,|b|)$
- $gcd(ka,kb) = k$ gcd(a,b)
- $gcd(a,b) = d \iff d|a, d|b$ and $gcd(a/d,b/d)=1$

It is possible to compute the greatest common divisor by means of the Euclidian algorithm shown in Algorithm 4.1.

Definition 4.5 (Prime numbers). *We say that a positive integer* $p > 1$ *is a prime number if its only positive divisors are* 1 *and* p.

Definition 4.6 (Relative Primes). *We say that two integers* a *and* b *are relatively primes if* $gcd(a,b)=1$.

Definition 4.7 (Composite Numbers). *If an integer number* $q > 1$ *is not a prime, then it is a composite number. Therefore, an integer* q *is a composite number if and only if there exist* a,b *positive integers (less than* q*) such that* $q = ab$.

Algorithm 4.1 Euclidean Algorithm (Computes the Greatest Common Divisor)

Require: two positive integers a and b where $a \geq b$.
Ensure: the greatest common divisor of a and b, namely $d = gcd(a, b)$.
1: **while** $b \neq 0$ **do**
2: $r \leftarrow a \bmod b$;
3: $a \leftarrow b$;
4: $b \leftarrow r$;
5: **end while**
6: **Return** a

Theorem 4.8 (Fundamental Theorem of Arithmetic). *Any natural number $n > 1$ is either a prime number, or it can be factored as a product of powers of prime numbers p_i,*

$$n = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$$

with $e_i \in \mathbb{N}$, $\forall\ i\ \in\ [\![1, r]\!]$. Furthermore, except for the order of the factors, this factorization is unique.

Corollary 4.9. *If $n \in \mathbb{N}$, then the number of positive divisors of n is $(e_1 + 1)(e_2 + 1) \cdots (e_r + 1)$.*

Corollary 4.10. *If p is a prime number, $a,\ b \in \mathbb{Z}$ and $p|ab$ then $p|a$ or $p|b$.*

Notice that above result is not necessarily true if p is a composite number. For example, $10|5 \cdot 4$ but $10 \not| 5$ and $10 \not| 4$.

Let $a,\ b \in \mathbb{N} \subset \mathbb{Z}$ and $a = \prod_{i=1}^{n} p_i^{e_i}$, and $b = \prod_{j=1}^{m} q_j^{f_j}$, be their prime factorization with $1 \leq i \leq n,\ 1 \leq j \leq m$. Let $R_1, R_2, \ldots, R_s$ be the distinct prime numbers that are included in both factorizations. Rewriting a and b as $a = \prod_{i=1}^{s} R_i^{t_i},\ b = \prod_{i=1}^{s} R_i^{u_i}$ with $t_i,\ u_i \geq 0$ for $1 \leq i \leq s$, we have,

$$gcd(a, b) = \prod_{i=1}^{s} R_i^{\min\{t_i, u_i\}}$$

Example 4.11.

$$2520 = 2^3 \cdot 3^2 \cdot 5^1 \cdot 7^1$$
$$2700 = 2^2 \cdot 3^3 \cdot 5^2 \cdot 7^0$$

then $gcd(2520, 2700) = 2^2 \cdot 3^2 \cdot 5^1 = 180$.

Definition 4.12. *Let $n \in \mathbb{N}$. We define the Euler function $\phi(n)$, as the number of relatively prime numbers that n has in the interval $[1, n)$.*

In other words, $\phi(n) = |\{m \in \mathbb{N} : gcd(m, n) = 1 \text{ and } 1 \leq m < n\}|$. Let p be a prime number and $m,\ n,\ r \in \mathbb{N}$ with $r > 1$, then

i. $\phi(p^r) = p^r\left(1 - \frac{1}{p}\right) = p^{r-1}(p-1)$, In particular $\phi(p) = p - 1$,
ii. $\phi(mn) = \phi(m)\phi(n)$, if $gcd(m, n) = 1$.

Therefore, we may compute the Euler function ϕ for a given number n by obtaining first the integer factorization of n.

Example 4.13.

$$\phi(720) = \phi(2^4)\phi(3^2)\phi(5) = 2^3 \cdot (2-1) \cdot 3^1 \cdot (3-1) \cdot (5-1) = 192.$$

Theorem 4.14 (Fermat's Little Theorem). *If $(a, p) = 1$, then*

$$a^{p-1} \equiv 1 \bmod p, \quad (a^p \equiv b \bmod p)$$

equivalently,

$$a^{\phi(p)} \equiv 1 \bmod p.$$

Corollary 4.15. *If $x \equiv y \mod (p-1)$, then $a^x \equiv a^y \bmod p$.*

Theorem 4.16 (Euler Theorem). *If $a \in \mathbb{Z}$ and gcd(m,a)=1 then*

$$a^{\phi(m)} \equiv 1 \bmod m.$$

Corollary 4.17. *If $x \equiv y \mod \phi(m)$, then $a^x \equiv a^y \bmod m$.*

Definition 4.18 (Order of a number x). *If x and m are relatively primes, we say that the order of x modulo m is the smallest integer r such that*

$$a^r = 1 \bmod m.$$

Definition 4.19 (Primitive Root). *Let m be a prime number and $g \in \mathbb{Z}_m$, then we say that g is a primitive root of m, if and only if the order of g modulo m is equal to the value of the Euler function $\phi(m)$. According to Euler's theorem, there is always a primitive root since, $g^{\phi(m)} = 1 \bmod m$.*

Let g be a primitive root of a prime number p, then the following properties hold,

i. If n is an integer, then $g^n \equiv 1 \bmod p$ if and only if $n \equiv 0 \bmod p - 1$.
ii. If j and k are two integers, then $g^j \equiv g^k \bmod p$ if and only if $j \equiv k \bmod p - 1$.
iii. If a is a primitive root, then a^x is also a primitive root if and only if $gcd(x, p-1) = 1$.

iv. If $g^n \equiv 1 \bmod p$ then $n|(p-1)$.

If $p = 1223$, $p-1 = 2 \cdot 13 \cdot 47$, if a is not a primitive root, then either a^{26} or a^{94} or a^{611} must be congruent 1 modulo 1223. $a = 2, 3$ are not primitive roots, since $2^{611} \equiv 3^{94} \equiv 1 \bmod 1223$. However, $a = 5$ is a primitive root since,

$$a^{26},\ a^{94}, a^{611} \neq 1 \bmod\ 1223.$$

Furthermore, using above properties we can see that $5^2 = 25$ is not a primitive root since $gcd(2, p-1) \neq 1$. On the other hand, the element $5^3 = 125$ is a primitive root given that $gcd(3, p-1) = 1$.

4.1.2 Modular Arithmetic

Definition 4.20 (Congruency). *Given $m \in \mathbb{Z}$, $m > 1$, we say that $a, b \in \mathbb{Z}$ are congruent modulo m if and only if $m|(a-b)$. We write this relation as $a \equiv b \bmod m$. Where m is the modulus of the congruency. Notice that if m divides $a-b$, this implies that both, a and b have the same residue when divided by m.*

We define $\mathbb{Z}_m$ as the set of all positive residues modulo m, which is composed by the set, $\mathbb{Z}_m = \{0, 1, 2, ..., m-1\}$. Invoking the integer division theorem it is easy to see that for every integer a there exists a residue r that belongs to $\mathbb{Z}_m$.

If $m \in \mathbb{N}$ and $a, b, c, d \in \mathbb{Z}$ such that $a \equiv b \bmod m$ and $c \equiv d \bmod m$, then the following properties hold,

- $a + c \equiv b + d \bmod m$
- $a - c \equiv b - d \bmod m$
- $a \cdot c \equiv b \cdot d \bmod m$

The relationship of congruency modulus m is a relationship of equivalence for all $m \in \mathbb{Z}$. Let $a, b, c \in \mathbb{Z}$, then the congruence relation satisfies the following properties,

1. Reflexive: $a \equiv a \bmod m$.
2. Symmetric: If $a \equiv b \bmod m$ then $b \equiv a \bmod m$.
3. Transitivity: If $a \equiv b \bmod m$ and $b \equiv c \bmod m$ then $a \equiv c \bmod m$.

Modular Addition and subtraction If $a, b \in \mathbb{Z}_m$ then we define the modular addition operator $a + b \bmod m$ as an element within $\mathbb{Z}_m$. For example, $17 + 20 \bmod 22 = 15$. The most important properties of the modular addition are,

1. It is commutative, $a + b \bmod m = b + a \bmod m$.
2. It is associative, $(a + b) + c \bmod m = a + (b + c) \bmod m$.
3. It has a neutral element (0), such that $a + 0 = a \bmod m$.

4. For every a and b in $\mathbb{Z}_m$ there exists a unique element x in $\mathbb{Z}_m$ such that $a + x = b \bmod m$.

Using last property and $b = 0$, it can be seen that for every a in $\mathbb{Z}_m$ there exists a unique element x in $\mathbb{Z}_m$ such that $a + x \equiv 0 \bmod m$.

Modular multiplication If $a, b \in \mathbb{Z}_m$ then we define modular multiplication as, $c = a \cdot b \bmod m$, where c is an element in $\mathbb{Z}_m$. The most important properties of modular multiplication are,

1. It is conmutative $a \cdot b \bmod m = b \cdot a \bmod m$.
2. It is associative $(a \cdot b) \cdot c \bmod m = a \cdot (b \cdot c) \bmod m$.
3. It has a neutral element (1), such that $a \cdot 1 = a \bmod m$
4. If $gcd(m, c)$=1 and $a \cdot c \equiv b \cdot c \bmod m$, then $a \equiv b \bmod m$. If m is a prime number, this property always hold.

Using last property, we define the *multiplicative inverse* of a number a as follows,

Definition 4.21 (Multiplicative Inverse). *We say that an integer a has an inverse modulo m if there exists an integer b such that $1 \equiv ab \bmod m$. Then, the integer b is the inverse of a and it is written as a^{-1}. The inverse of a number $a \bmod m$ exists if and only if there exist two integer numbers x, y such that $ax + my = 1$ and these numbers exist if and only if gcd(a,m)=1.*

In order to obtain the modular inverse of a number a we may use the extended Euclidean algorithm [178], with which it is possible to find the two integer numbers x, y that satisfy the equation[1],

$$ax + my = 1.$$

Modular Division Using above definition we say that if $a, b \in \mathbb{Z}_p$ and p is a prime number, we can accomplish the division of a by b by computing $a \cdot b^{-1} \bmod m$, where b^{-1} is the multiplicative inverse of b modulo p.

For example, we can compute $\frac{17}{20} \bmod 23$, by performing $17 \cdot (20)^{-1} \bmod 23$, where $(20)^{-1} \bmod 23 = 15$. Thus,

$$\frac{17}{20} \bmod 23 = 17 \cdot 15 \bmod 23 = 2.$$

Modular Exponentiation We define modular exponentiation, as the problem of computing the number $b = a^e \bmod m$, with $a, b \in \mathbb{Z}_m$, and $e \in \mathbb{N}$. From the observation that,

$$x \cdot y \bmod m = [(x \bmod m) \cdot y \bmod m] \bmod m,$$

[1] In §6.3 we present an efficient implementation of a variation of this algorithm: the Binary Euclidean Algorithm (BEA).

Algorithm 4.2 Extended Euclidean Algorithm as Reported in [228]

Require: Two positive integers a and b where $a \geq b$.
Ensure: d =gcd(a, b) and the two integers x,y that satisfy the equation $ax + by = d$.

```
1: if b = 0 then
2:    d = a;, x = 1;, y = 0;
3:    Return (d, x, y)
4: end if
5: x1 = 0;, x2 = 1;, y1 = 1;, y2 = 0;
6: while b > 0 do
7:    q = a div b; r = a mod b;
8:    x = x2 − qx1; y = y2 − qy1;
9:    a = b; b = r; x2 = x1;
10:   x1 = x; y2 = y1; y1 = y;
11: end while
12: d = a, x = x2, y = y2;
13: Return (d, x, y)
```

it can be seen that the exponentiation problem, can be solved by multiplying numbers that never exceed the modulus m.

Rather than computing the exponentiation by performing $e - 1$ modular multiplications as,

$$b = \overbrace{a \cdot a \ldots a}^{e-1 mults.} \pmod{m},$$

we employ a much more efficient method that has complexity $O(log(e))$. For example if we want to compute 12^{26}(mod23), we can proceed as follows,

$$\begin{aligned} 12^2 &= 144 = 6 \bmod 23; \\ 12^4 &= 62 = 36 = 13 \bmod 23; \\ 12^8 &= 132 = 169 = 8 \bmod 23; \\ 12^{16} &= 82 = 64 = 18 \bmod 23. \end{aligned}$$

Then,

$$12^{26} = 12^{(16+8+2)} = 12^{16} \cdot 12^8 \cdot 12^2 = 18 \cdot 8 \cdot 6 = 864 = 13 \bmod 23.$$

This algorithm is known as the binary exponentiation algorithm [178], whose details will be discussed in §5.4.

Chinese Remainder Theorem(CRT) This theorem has a tremendous importance in cryptography. It can be defined as follows,

Let p_i for $i = 1, 2, \ldots, k$ be pairwise relatively prime integers, i.e.,

$$gcd(p_i, p_j) = 1 \text{ for } i \neq j.$$

Given $u_i \in [0, p_i - 1]$ for $i = 1, 2, \ldots, k$, the Chinese remainder theorem states that there exists a unique integer u in the range $[0, P-1]$ where $P = p_1 p_2 \cdots p_k$ such that

$$u = u_i \pmod{p_i}.$$

4.2 Finite Fields

We start with some basic definitions and then arithmetic operations for the finite fields are explained.

4.2.1 Rings

A ring $\mathbb{R}$ is a set whose objects can be added and multiplied, satisfying the following conditions:

- Under addition, $\mathbb{R}$ is an additive (Abelian) group.
- For all $x; y; z \in \mathbb{R}$ we have, $x(y + z) = xy + xz$; $(y + z)x = yx + zx$:
- For all $x; y \in \mathbb{R}$, we have $(xy)z = x(yz)$.
- There exists an element $e \in \mathbb{R}$ such that $ex = xe = x$ for all $x \in \mathbb{R}$.

The integer numbers, the rational numbers, the real numbers and the complex numbers are all rings. An element x of a ring is said to be invertible if x has a multiplicative inverse in $\mathbb{R}$, that is, if there is a unique $u \in \mathbb{R}$ such that: $xu = ux = 1$. 1 is called the *unit element* of the ring.

4.2.2 Fields

A Field is a ring in which the multiplication is commutative and every element except 0 has a multiplicative inverse. We can define a Field $\mathbb{F}$ with respect to the addition and the multiplication if:

- $\mathbb{F}$ is a commutative group with respect to the addition.
- $\mathbb{F} \setminus \{0\}$ is a commutative group with respect to the multiplication.
- The distributive laws mentioned for rings hold.

4.2.3 Finite Fields

A *finite field* or *Galois field* denoted by $\mathrm{GF}(q = p^m)$, is a field with characteristic p, and a number q of elements. Such a finite field exists for every prime p and positive integer m, and contains a subfield having p elements. This subfield is called *ground field* of the original field. For every non-zero element $\alpha \in \mathrm{GF}(q)$, the identity $\alpha^{q-1} = 1$ holds.

In cryptography the two most studied cases are: $q = p$, with p a prime and $q = 2^m$. The former case, $\mathrm{GF}(p)$, is denoted as *prime field*, whereas the latter, $\mathrm{GF}(2^m)$, is known as finite field of characteristic two or simply *binary extension field*. A binary extension field is also denoted as $\mathbb{F}_{2^m}$.

4.2.4 Binary Finite Fields

A polynomial p in $GF(q)$ is *irreducible* if p is not a unit element and if $p = fg$ then f or g must be a unit, that is, a constant polynomial.

Let $P(x)$ be an irreducible polynomial over $GF(2)$ of degree m, and let α be a root of $P(x)$, i.e., $P(\alpha) = 0$. Then, we can use $P(x)$ to construct a binary finite field $F = GF(2^m)$ with exactly $q = 2^m$ elements, where α itself is one of those elements. Furthermore, the set

$$\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$$

forms a basis for F, and is called the polynomial (canonical) basis of the field [221]. Any arbitrary element $A \in GF(2^m)$ can be expressed in this basis as,

$$A = \sum_{i=0}^{m-1} a_i \alpha^i.$$

Notice that all the elements in F can be represented as $(m-1)$-degree polynomials.

The order of an element $\gamma \in F$ is defined as the smallest positive integer k such that $\gamma^k = 1$. Any finite field contains always at least one element, called a *primitive element*, which has order $q-1$. We say that $P(x)$ is a primitive polynomial if any of its roots is a primitive element in F. If $P(x)$ is primitive, then all the q elements of F can be expressed as the union of the zero element and the set of the first $q-1$ powers of α [221, 379]

$$\left\{0, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{q-1} = 1\right\}. \tag{4.1}$$

Some special classes of irreducible polynomials are more convenient for the implementation of efficient binary finite field arithmetic. Some important examples are: trinomials, pentanomials, and equally-spaced polynomials. Trinomials are polynomials with three non-zero coefficients of the form,

$$P(x) = x^k + x^n + 1 \tag{4.2}$$

Whereas pentanomials have five non-zero coefficients:

$$P(x) = x^k + x^{n_2} + x^{n_1} + x^{n_0} + 1 \tag{4.3}$$

Finally, irreducible equally-spaced polynomials have the same space separation between two consecutive non-zero coefficients. They can be defined as

$$P(x) = x^m + x^{(k-1)d} + \cdots + x^{2d} + x^d + 1\ , \tag{4.4}$$

where $m = kd$. The ESP specializes to the all-one-polynomials (AOPs) when $d = 1$, i.e., $P(x) = x^m + x^{m-1} + \cdots + x + 1$, and to the equally-spaced trinomials when $d = \frac{m}{2}$, i.e., $P(x) = x^m + x^{\frac{m}{2}} + 1$.

In this Book we are mostly interested in a *polynomial basis* representation of the elements of the binary finite fields. We represent each element as a binary string $(a_{m-1} \ldots a_2 a_1 a_0)$, which is equivalently considered a polynomial of degree less than m,

$$a_{m-1}x^{m-1} + \ldots + a_2 x^2 + a_1 x + a_0. \tag{4.5}$$

The addition of two elements $a, b \in F$ is simply the addition of two polynomials, where the coefficients are added in $GF(2)$, or equivalently, the bitwise XOR operation on the vectors a and b. Multiplication is defined as the polynomial product of the two operands followed by a reduction modulo the generating polynomial $p(x)$. Finally, the inversion of an element $a \in F$ is the process to find an element $a^{-1} \in F$ such that $a \cdot a^{-1} = \mod P(x)$.

Addition is by far the less costly field operation. Thus, its computational complexity is usually neglected (i.e., considered 0). Inversion, on the other hand, is considered the most costly field operation.

Example 4.22. The sum of the two polynomials A and B, denoted in hexadecimal representation as 57 and 83, respectively, is the polynomial denoted by D4, since:

$$\begin{aligned}
&(x^6 + x^4 + x^2 + x + 1) \oplus (x^7 + x + 1) \\
&= x^7 + x^6 + x^4 + x^2 + (1 \oplus 1)x + (1 \oplus 1) \\
&= x^7 + x^6 + x^4 + x^2
\end{aligned}$$

In binary notation we have: $01010111 \oplus 10000011 = 11010100$. Clearly, the addition can be implemented with the bitwise XOR instruction.

Example 4.23. Let us consider the irreducible pentanomial P(x), defined as,

$$P(x) = x^8 + x^4 + x^3 + x + 1 \tag{4.6}$$

Since P(x) is irreducible over $GF(2)$, we have constructed a representation for the field GF(2^8). Hence we can say that byte chains can be considered as elements of GF(2^8). For example, consider the multiplication of the field elements $A = (57)_{16}$ and $B = (83)_{16}$. The resulting field product, $C = AB \mod P(x)$, is $C = (C1)_{16}$, since,

$$\begin{aligned}
&(x^6 + x^4 + x^2 + x + 1) \times (x^7 + x + 1) \\
&= (x^{13} + x^{11} + x^9 + x^8 + x^7) \oplus (x^7 + x^5 + x^3 + x^2 + x) \\
&\quad \oplus (x^6 + x^4 + x^2 + x + 1) \\
&= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1
\end{aligned}$$

and

$$\begin{aligned}
&(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \\
&\equiv x^7 + x^6 + 1 \bmod (x^8 + x^4 + x^3 + x + 1)
\end{aligned}$$

4.3 Elliptic curves

The theory of elliptic curves has been studied extensively in number theory and algebra for the past 150 years. It has been developed a rich and deep theoretical background initially tailored for purely aesthetic reasons. Elliptic curve cryptosystems were proposed for the first time by N. Koblitz [180] and V. Miller [236]. Since then a vast amount of literature has been accumulated on this topic. Recently elliptic curve cryptosystems are widely accepted for security applications like key generation, signature and verification.

Elliptic curves can be defined over real numbers, complex numbers and any other field. In order to explain the geometric properties of elliptic curves let us first examine elliptic curves defined over the real numbers $\mathbb{R}$.

Nonetheless, we stress that elliptic curves over finite fields are the only relevant ones from the cryptographic point of view. More specifically binary representation of elliptic curves will be discussed here which is directly related to the work to be presented in Chapter 10.

In the rest of this section, basic definitions and common operations of elliptic curves will be explained.

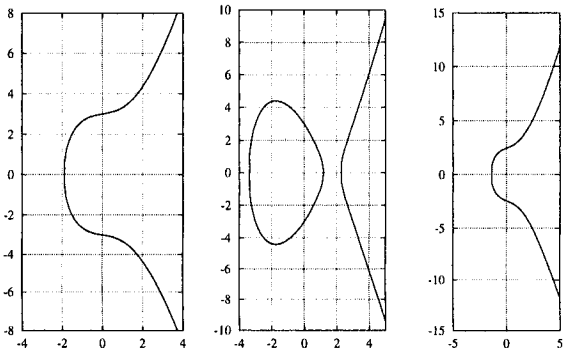

Fig. 4.1. Elliptic Curve Equation $y^2 = x^3 + ax + b$ for Different a and b

4.3.1 Definition

Elliptic curves over real numbers are defined as the set of points (x, y) which satisfy the elliptic curve equation of the form:

$$y^2 = x^3 + ax + b \tag{4.7}$$

where a and b are real numbers. Each choice of a and b produces a different elliptic curve as shown in Figure 4.1. The elliptic curve in Equation 4.7 forms a group if $4a^3 + 27b^2 \neq 0$. An elliptic curve group over real numbers consists of the points on the corresponding elliptic curve, together with a special point $\mathcal{O}$ called the point at infinity.

4.3.2 Elliptic Curve Operations

Elliptic curve groups are additive groups; that is, their basic function is addition. To visualize the addition of two points on the curve, a geometric representation is preferred. We define the negative of a point $P = (x, y)$ as its reflection in the x-axis: the point $-P$ is $(x, -y)$. Also if the point P is on the curve, the point $-P$ is also on the curve.

In the rest of this subsection the addition operation for two distinct points on the curve are explained. Some special cases for the addition of two points on the curve are also described.

- **Adding distinct P and Q**: Let P and Q be two distinct points on an elliptic curve, and $P \neq -Q$. The addition law in an elliptic curve group is $P + Q = R$. For the addition of the points P and Q, a line is drawn through the two points that will intersect the curve at another point, call $-R$. The point $-R$ is reflected in the x-axis to get a point R which is the required point. A geometrical representation of adding two distinct points on the elliptic curve is shown in Figure 4.2.

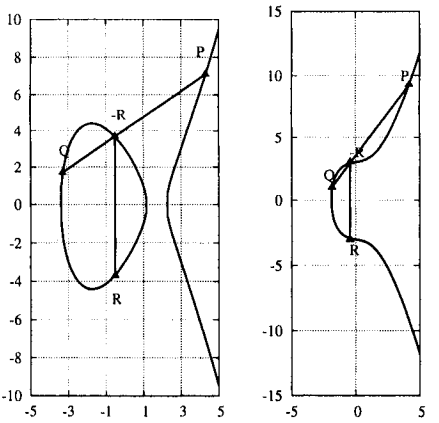

Fig. 4.2. Adding two Distinct Points on an Elliptic curve ($Q \neq -P$)

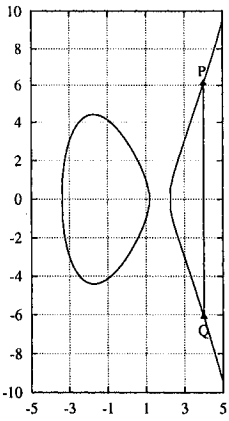

Fig. 4.3. Adding two Points P and Q when $Q = -P$

- **Adding P and $-P$:** The method for adding two distinct points P and Q cannot be adopted for the addition of the points P and $-P$ because the line through P and $-P$ is a vertical line which does not intersect the elliptic curve at a third point as shown in Figure 4.3. This is the reason why the elliptic curve group includes the point at infinity $\mathcal{O}$. By definition, $P+(-P) = \mathcal{O}$. As a result of this equation, $P+\mathcal{O} = \mathcal{P}$ in the elliptic curve group. The point at infinity $\mathcal{O}$ is called the additive identity of the elliptic curve group. All well-defined elliptic curves have an additive identity.

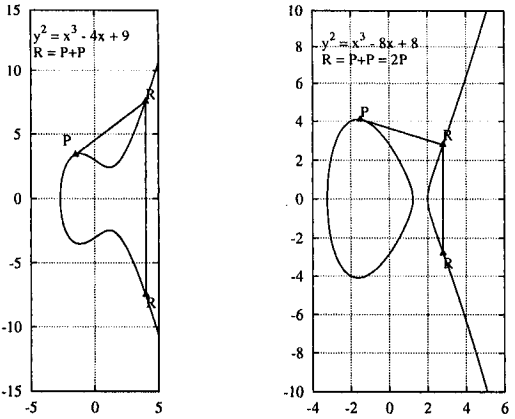

Fig. 4.4. Doubling a Point P on an Elliptic Curve

- **Doubling $P(x, y)$ when $y \neq 0$:**

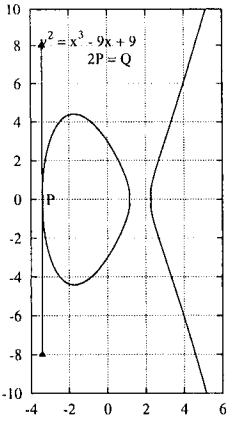

Fig. 4.5. Doubling $P(x, y)$ when $y = 0$

The law for doubling a point on an elliptic curve group is defined by: $P + P = 2P = R$. To add a point $P(x, y)$ to itself, a tangent line to the curve is drawn at the point P. If $y \neq 0$, then the tangent line intersects the elliptic curve at exactly one other point $-R$ as shown in Figure 4.4. The point $-R$ is reflected in the x-axis to R which is the required point. This operation is called doubling the point P.

- **Doubling $P(x, y)$ when $y = 0$:** If for a point $P(x, y)$, $y = 0$, then it does not intersect the elliptic curve at any other point because the tangent line to the elliptic curve at P is vertical. By definition, $2P = \mathcal{O}$ for such a point P. If one wants to find $3P$ in this situation, one can add $2P + P$. This becomes $P + \mathcal{O} = \mathcal{P}$. Thus $3P = P$, $4P = \mathcal{O}$, $5P = P$, $6P = \mathcal{O}$, $7P = P$, etc.

4.3.3 Elliptic Curve Scalar Multiplication

There is no multiplication operation in elliptic curve groups. However, the scalar product kP can be obtained by adding k copies of the same point P, which can be accomplished using the addition and doubling operations explained in the last Subsection. Thus the product $kP = P + P + ...P$ obtained in this way is referred to elliptic curve scalar multiplication. Figure 4.6 shows the scalar multiplication process for obtaining 6 copies of the point P. However for professional elliptic curve cryptosystem implementations, much higher values of k are used. Typically, the bit-length of k is selected in the range of 160-521 bits.

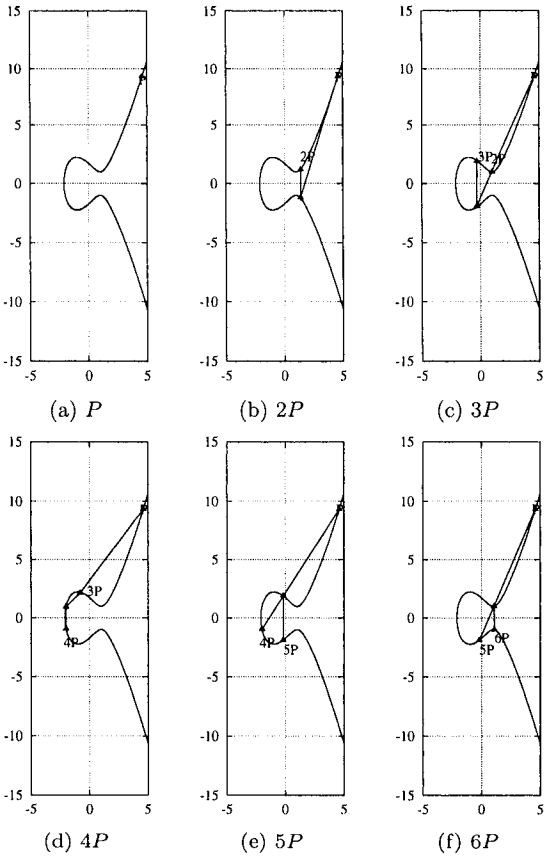

Fig. 4.6. Elliptic Curve Scalar Multiplication kP, for $k = 6$ and for the Elliptic Curve $y^2 = x^3 - 3x + 3$

4.4 Elliptic Curves over $GF(2^m)$

Because of the chracteristic two, the equation for the elliptic curve with the underlying field $GF(2^m)$ is slightly adjusted as shown in Equation 4.8. It is formed by choosing the elements a and b within $GF(2^m)$ with $b \neq 0$.

$$y^2 + xy = x^3 + ax^2 + b \tag{4.8}$$

The elliptic curve includes all points (x, y) which satisfy the elliptic curve equation over $GF(2^m)$ (where x and $y \in GF(2^m)$). An elliptic curve group over

$GF(2^m)$ consists of the points on the corresponding elliptic curve, together with a point at infinity, $\mathcal{O}$.

The points on an elliptic curve can be represented using either two or three coordinates. In affine-coordinate representation, a finite point on $E(GF(2^m))$ is specified by two coordinates x; $y \in GF(2^m)$ satisfying Equation 4.8. The point at infinity has no affine coordinates.

We can make use of the concept of a projective plane over the field $GF(2^m)$ [228]. In this way, one can represent a point using three rather than two coordinates. Then, given a point P with affine-coordinate representation x; y; there exists a corresponding projective-coordinate representation X; Y and Z such that,

$$P(x; y) \equiv P(X; Y; Z)$$

The formulae for converting from affine coordinates to Jacobian projective coordinates and vice versa are given as:

$$\begin{array}{l} \text{Affine-to-Projective: } X = x; \quad Y = y; \quad Z{=}1 \\ \text{Projective-to-Affine: } x = X/Z^2; \; y = Y/Z^3 \end{array}$$

The algebraic formulae for the group law are different for affine and projective coordinates. In the next subsections the group law over $GF(2^m)$ is explained using affine coordinates representation. The group laws for several projective coordinates representations are studied in §4.5.

4.4.1 Point Addition

The negative of a point $P = (x, y)$ is $-P = (x, x + y)$. Assuming that $P \neq Q$, then $R(x_3, y_3) = P(x_1, y_1) + Q(x_2, y_2)$ where:

$$\begin{array}{l} m = \frac{(y_2+y_1)}{(x_2+x_1)} \\ x_3 = m^2 + m + x_1 + x_2 + a \\ y_3 = m(x_1 + x_3) + x_3 + y_1 \end{array} \tag{4.9}$$

As with elliptic curve groups over real numbers, $P + (-P) = \mathcal{O}$, where $\mathcal{O}$ the point at infinity. Furthermore, $P + \mathcal{O} = \mathcal{P}$ for all points P in the elliptic curve group.

4.4.2 Point Doubling

Let $P(x_1, y_1)$ be a point on the curve. If $x_1 = 0$, then $2P = \mathcal{O}$. If $x_1 \neq 0$ then $R = 2P$, and $R(x_2, y_2)$ is given as:

$$\begin{array}{l} x_2 = x_1^2 + \frac{b}{x_1^2} \\ y_2 = x_1^2 + (x_1 + \frac{y_1}{x_1})x_2 + x_2 \end{array} \tag{4.10}$$

Let us recall that a is one of the parameters chosen with the elliptic curve and that m is the slope of the line through P and Q.

4.4.3 Order of an Elliptic Curve

Notice that the elliptic curve $E(\mathbb{F}_q)$, namely the collection of all the points in $\mathbb{F}_q$ that satisfy Eq. (4.10) can only be finitely many. Even if every possible pair (x, y) were on the curve, there would be only q^2 possibilities. As a matter of fact, the curve $E(\mathbb{F}_q)$ could have at most $2q+1$ points because we have one point at infinity and $2q$ pairs (x, y) (for each x we have two values of y).

The total number of points in the curve, including the point $\mathcal{O}$, is called the *order* of the curve. The order is written $\#E(\mathbb{F}_q)$. A celebrated result discovered by Hasse gives the lower and the upper bounds for this number.

Theorem 4.24. *[227] Let $\#E(\mathbb{F}_q)$ be the number of points in $E(\mathbb{F}_q)$. Then,*

$$|\#E(\mathbb{F}_q) - (q+1)| \leq 2\sqrt{q} \tag{4.11}$$

The interval $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$ is called the Hasse interval.

As we did in the case of finite fields, we can also introduce the concept of the order of an element in elliptic curves. The order of a point P on $E(\mathbb{F}_q)$ is the smallest integer n such that $nP = 0$. The order of any point it is always defined, and divides the order of the curve $\#E(\mathbb{F}_q)$. This guarantees that if r and l are integers, then $rP = lP$ if and only if $r \equiv l \pmod{n}$.

4.4.4 Elliptic Curve Groups and the Discrete Logarithm Problem

Every cryptosystem is based on a hard mathematical problem that is computationally infeasible to solve. The discrete logarithm problem is the basis for the security of many cryptosystems including Elliptic Curve Cryptosystems. More specifically the security of elliptic curve cryptosystems relies on Elliptic Curve Discrete Logarithmic Problem (ECDLP).

In the last Section we examined two elliptic curve operations: point addition and point doubling. Both point addition and doubling operations can be used to compute any number of copies of a point ($2P$, $3P$, kP, etc). The determination of a point kP in this manner is referred to as *Scalar Multiplication* of a point. In the rest of this Section we present a small example of how to compute such elliptic curve operation.

4.4.5 An Example

Let $F = GF(2^4)$ be a binary finite field with defining primitive trinomial $p(x)$ given as,

$$p(x) = x^4 + x + 1. \tag{4.12}$$

Then, if α is a root of $p(x)$, we have $p(\alpha) = 0$, which implies,

$$p(\alpha) = \alpha^4 + \alpha + 1 = 0. \tag{4.13}$$

For binary field arithmetic, addition is equivalent to subtraction. Hence, the above equation can be rewritten as

$$\alpha^4 = \alpha + 1. \tag{4.14}$$

Using equation (4.14), one can now express each one of the 15 nonzero elements of F as is shown in Table 4.1. Notice that we can define any one of the $q = 2^4$ elements of F using only four coordinates.

Element in $GF(2^m)$	Polynomial	Coordinates
0	0	(0000)
α	α	(0010)
α^2	α^2	(0100)
α^3	α^3	(1000)
α^4	$\alpha + 1$	(0011)
α^5	$\alpha^2 + \alpha$	(0110)
α^6	$\alpha^3 + \alpha^2$	(1100)
α^7	$\alpha^3 + \alpha + 1$	(1011)
α^8	$\alpha^2 + 1$	(0101)
α^9	$\alpha^3 + \alpha$	(1010)
α^{10}	$\alpha^2 + \alpha + 1$	(0111)
α^{11}	$\alpha^3 + \alpha^2 + \alpha$	(1110)
α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$	(1111)
α^{13}	$\alpha^3 + \alpha^2 + 1$	(1101)
α^{14}	$\alpha^3 + 1$	(1001)
α^{15}	1	(0001)

Table 4.1. Elements of the field $F = GF(2^4)$, Defined Using the Primitive Trinomial of Eq. ((4.12))

Notice that all the elements in F can be described by any of the three representations used in Table 4.1, namely, polynomial representation, coordinate representation and powers of the primitive element α.

Let us now consider a non-supersingular elliptic curve defined as the set of points $(x, y) \in F \times F$ that satisfy

$$y^2 + xy = x^3 + \alpha^{13}x^2 + \alpha^6 \tag{4.15}$$

Notice that for the coefficients a and b of equation (4.8), we have selected the values α^{13} and α^6, respectively. There exist a total of 14 solutions in such a curve, including the point at infinite $\mathcal{O}$. Using table 4.1, we can see that, for example, the point,

$$P = (x_p, y_p) = (\alpha^3, \alpha^2) \tag{4.16}$$

satisfies equation (4.15) over $\mathbb{F}_2^4$, since

$$\begin{array}{ll} y^2 + xy & = x^3 + \alpha^{13}x^2 + \alpha^6 \\ (\alpha^2)^2 + \alpha^3\alpha^2 & = (\alpha^3)^3 + \alpha^{13}(\alpha^3)^2 + \alpha^6 \\ \alpha^4 + \alpha^5 & = \alpha^9 + \alpha^{19} + \alpha^6 \\ & = \alpha^9 + \alpha^4 + \alpha^6 \\ (0011) + (0110) & = (1010) + (0011) + (1100) \\ (0101) & = (0101), \end{array} \tag{4.17}$$

Where we have used the identity $\alpha^{15} = 1$. All the thirteen finite points which satisfy equation (4.15) are shown in figure 4.7.

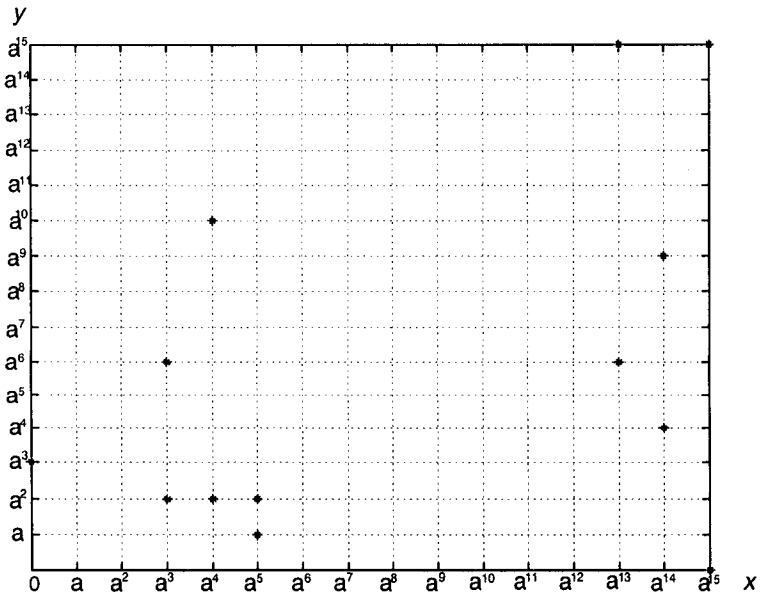

Fig. 4.7. Elements in the Elliptic Curve of Equation (4.15)

Let us now use equation (4.10) to double the point $P = (\alpha^3, \alpha^2)$. Using once again table 4.1, we obtain,

$$\begin{aligned} x_{2p} &= x_p^2 + \tfrac{b}{x_p^2} \\ &= (\alpha^3)^2 + \alpha^6 \cdot (\alpha^3)^{-2} \\ &= \alpha^6 + \alpha^6 \cdot \alpha^{-6} = \alpha^6 + 1 = \alpha^{13} \\ y_{2p} &= x_p^2 + \left(x_p + \tfrac{y_p}{x_p}\right) x_{2p} + x_{2p} \\ &= \alpha^6 + \left(\alpha^3 + \alpha^2 \cdot \alpha^{-3}\right) \alpha^{13} + \alpha^{13} \\ &= \alpha^6 + \left(\alpha^3 + \alpha^{-1}\right) \alpha^{13} + \alpha^{13} \\ &= \alpha^6 + \alpha^1 + \alpha^{12} + \alpha^{13} = \alpha^6 \end{aligned} \tag{4.18}$$

It can be verified from figure 4.7 that the result obtained above is indeed a point in the elliptic curve of equation (4.15).

As we mentioned in §4.4.3, we can keep adding P to its scalar multiples, but eventually, after $n \leq \#E(\mathbb{F}_q)$ scalar multiplications, we will obtain the point at infinite $\mathcal{O}$ as a result. Recall that the integer n is called the order of the point P. For the case in hand, P happens to have a prime order $k = 7$. Notice that as it was stated in §4.4.3, the order n of P divides the order of the curve $\#E(\mathbb{F}_q)$. Table 4.2 lists all the six finite multiples of P.

P	$2P$	$3P$	$4P$	$5P$	$6P$
(α^3, α^2)	(α^{13}, α^6)	(α^{14}, α^9)	(α^{14}, α^4)	$(\alpha^{13}, \alpha^{15})$	(α^3, α^6)

Table 4.2. Scalar Multiples of the Point P of Equation (4.16)

Obviously, in a true cryptographic application the parameter n should be chosen large enough so that efficient generation of such a look-up table approach, becomes unfeasible. In today's practice, $n \geq 2^{160}$ has proved to be sufficient.

4.5 Point Representation

In order to generate an Abelian group over elliptic curves, it was necessary to define an elliptic curve group law. More specifically, we defined the point addition and point doubling primitives of Equations (4.9) and (4.10). However, the computational cost of those equations involves the calculation of a costly field inverse operation plus several field multiplications.

Since the relation (I/M) defined as the computational cost of a field inversion over the computational cost of a field multiplication is above 8 and 20 in hardware and software implementations, respectively, there is a strong motivation for finding alternative point representations that allow the trading of the costly field inversions by less expensive field multiplications.

As we have seen at the beginning in §4.4, elliptic point representation in two coordinates is called *affine representation*, whereas the equivalent point representation in three coordinates is called *Projective representation.*

It can be shown that each affine point can be related one-to-one with a unique equivalence class. Then, each elliptic point is represented by a triple that satisfy the corresponding equivalence class. Notice that it results necessary to redefine the addition and doubling operations in the projective representation.

As it will be explained in the rest of this Section, the projective group law can be implemented without utilizing field inversions at the price of increasing the total number of field multiplications. As a matter of fact, field inversions are only required when converting from projective representation to affine representation[2], which becomes valuable in situations where we are planning to perform many point additions and doublings in a successive manner (such as in elliptic curve scalar multiplication).

4.5.1 Projective Coordinates

Let c and d be positive integers over the field K. It is possible to define an equivalent class $K^3 \setminus \{(0,0,0)\}$ as follows,

$$(X_1, Y_1, Z_1) \sim (X2, Y2, Z2) | \text{ If } X_1 = \lambda^c X_2, Y_1 = \lambda^d Y_2, Z_1 = \lambda Z_2.$$

The equivalent class

$$(X : Y : Z) = \left\{(\lambda^c X, \lambda^d Y, \lambda Z) : \lambda \in K^*\right\}.$$

is called a *projective point* [129], and (X, Y, Z) a *representative point* of such class, that is to say, any point within the class is a representative point. Specifically, if $Z \neq 0$, $(\frac{X}{Z^c}, \frac{Y}{Z^d}, 1)$ is a point representative of the equivalence class $(X : Y : Z)$.

Therefore, if we define the set of all projective points (equivalent classes) for each possible λ in the field K^* as,

$$P(K)^* = \{(X : Y : Z) : X, Y, Z \in K, Z \neq 0\},$$

we obtain a one-to-one correspondence between the point $P(K)^*$ and the set of affine points,

$$A(K) = \{(x, y : x, y \in K)\}.$$

Each point in the *affine coordinate system*, corresponds to the set defined by an equivalence class in particular. The set of point belonging to $P(K)^0 = \{(X : Y : Z) : X, Y, Z \in K, Z = 0\}$ is called the *line at infinity*, because this class does not correspond with any element in the set of affine points.

[2] In §4.4 the explicit conversion equations from affine to Jacobian projective coordinates and vice versa were stated.

The Weierstrass equation for an elliptic curve $E(K)$ can be defined in projective coordinates by replacing x by $\frac{X}{Z^c}$ and y by $\frac{Y}{Z^d}$. The constant values c and d will determine the characteristic of the elliptic curve arithmetic and hence, the definition of the point addition algorithm in such representation.

4.5.2 López-Dahab Coordinates

The most popular projective coordinate system are the *standard* where $c = 1$ and $d = 1$, Jacobians, with $c = 2$ and $d = 3$ and López-Dahab (LD) coordinates, , with $c = 1$ and $d = 2$. The latter system of coordinates offers algorithms for computing the addition in *mixed coordinates*, i.e., one point is given in affine coordinates while the other is given in projective coordinates. LD coordinates are highly attractive for hardware implementation because they only employ 8 field multiplications for performing a point addition operation.

In *López-Dahab (LD)* projective coordinates [210] the projective point (X: Y: Z) with Z$\neq$ 0 corresponds to the affine coordinates $x = X/Z$ and $y = Y/Z^2$. Therefore, the elliptic curve equation (4.8) mapped to LD projective coordinates can be written as,

$$Y^2 + XYZ = X^3Z + aX^2Z^2 + Z^4 \tag{4.19}$$

The point at infinity is represented now as $\mathcal{O} = (1 : 0 : 0)$. For any arbitrary point P on the curve, it holds that $P+\mathcal{O} = \mathcal{O}+\mathcal{P} = \mathcal{P}$. Let $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : 1)$ be two arbitrary points belonging to the curve 4.19. Then the point $-P = (X_1 : X_1 + Y_1 : Z)$ is the addition inverse of the point P. The point doubling primitive $2(X_1 : Y_1 : Z_1) = (X_3 : Y_3 : Z_3)$ can be performed at a computational cost of 2 general field multiplications plus two field multiplication by the elliptic curve constant b as [212],

$$\begin{aligned} Z_3 &= X_1^2 \cdot Z_1^2, \\ X_3 &= X_1^4 + b \cdot Z_1^4, \\ Y_3 &= bZ_1{}^4 Z_3 + X_3 \cdot (aZ_3 + Y_1{}^2 + bZ_1{}^4) \end{aligned} \tag{4.20}$$

Whereas if $Q \neq -P$, the point addition primitive $(X_1 : Y_1 : Z_1) + (X_2 : Y_2) = (X_3 : Y_3 : Z_3)$ can be performed at a computational cost of 8 field multiplications as,

$$\begin{aligned} A &= Y_2 \cdot Z_1^2 + Y_1; & B &= X_2 \cdot Z_1 + X_1; \\ C &= Z_1 \cdot B; & D &= B^2 \cdot (C + aZ_1^2); \\ Z_3 &= C^2; & E &= A \cdot C; \\ X_3 &= A^2 + D + E; & F &= X_3 + X_2 \cdot Z_3; \\ G &= (X_2 + Y_2) \cdot Z_3^2; & Y_3 &= (E + Z_3) \cdot F + G \end{aligned} \tag{4.21}$$

4.6 Scalar Representation

The vast majority of algorithms reported for computing the scalar multiplication in an efficient manner are based in the Horner polynomial representation,

$$a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0 = a_0 + (a_1 + (a_2 + (\ldots + (a_{n-1} + (a_n + x)x)\ldots)x)x)x.$$

where the scalar k is represented using its binary expansion, namely, $k = b_n 2^n + b_{n-1} + 2^{n-1} + \ldots + b_1 2 + b_0$ where $b_i \in [0, 1]$.

4.6.1 Binary Representation

Algorithm 4.3 Basic Doubling & Add algorithm for Scalar Multiplication

Require: $k = (k_{m-1}, k_{m-2}, \ldots, k_1, k_0)_2$ with $k_{n-1} = 1$, $P(x, y, z) \in E(\mathbb{F}_{2^m})$
Ensure: $Q = kP$
1: $Q = P$;
2: **for** $i = m - 2$ **downto** 0 **do**
3: $Q = 2 \cdot Q$ (point doubling) ;
4: **if** $k_i = 1$ **then**
5: $Q = Q + P$ (point addition);
6: **end if**
7: **end for**
8: **Return Q**

The traditional method for computing the elliptic operation kP is based in the binary representation of k. If $k = \sum_{j=0}^{m-1} b_j 2^j$, where each $b_j \in \{0, 1\}$, then kP can be computed as [227]:

$$kP = \sum_{j=0}^{m-1} b_j 2^j P = 2(\ldots 2(2b_{m-1}P + b_{m-2}P) + \ldots) + b_0 P.$$

This method requires $m-1$ point doublings and $w_k - 1$ point additions, where w_k is the Hamming weight (total number of coefficients $b_j = 1$) of the binary representation of the scalar k.

4.6.2 Recoding Methods

It is possible to reduce the number of subsequent point additions using a *recoding* of the the exponent [154, 239, 76, 176]. The recoding techniques use the identity

$$2^{i+j-1} + 2^{i+j-2} + \cdots + 2^i = 2^{i+j} - 2^i$$

to collapse a block of 1s in order to obtain a *sparse* representation of the exponent. Thus, a redundant signed-digit representation of the exponent using the digits $\{0, 1, -1\}$ will be obtained. For example, (011110) can be recoded as

Algorithm 4.4 The Recoding Binary algorithm for Scalar Multiplication

Require: $k = (k_{m-1}, k_{m-2}, \ldots, k_1, k_0)_2$ with $k_i \in [\![-1, 0, 1]\!]$, $P(x, y, z) \in E(\mathbb{F}_{2^m})$
Ensure: $Q = kP$
1: $Q = P$;
2: **for** $i = m - 2$ **downto** 0 **do**
3: $Q = 2 \cdot Q$ (point doubling) ;
4: **if** $k_i = 1$ **then**
5: $Q = Q + P$ (point addition);
6: **else if** $k_i = \bar{1}$ **then**
7: $Q = Q - P$ (point subtraction);
8: **end if**
9: **end for**
10: **Return Q**

$$(011110) = 2^4 + 2^3 + 2^2 + 2^1$$
$$(1000\bar{1}0) = 2^5 - 2^1.$$

The recoding binary method is given in the Algorithm 4.4. Note that even though the number of bits of k is equal to m, the number of bits in the recoded exponent $\hat{k}$ can be $m + 1$, for example, (111) is recoded as $(100\bar{1})$. Thus, the recoding binary algorithm starts from the bit position m in order to compute kP by computing $\hat{k}P$ where $\hat{k}$ is the $(k + 1)$-bit recoded exponent such that $\hat{k} = k$.

Let us discuss an explicit toy example of scalar multiplication using the recoding binary method. Let $k = 119 = (1110111)$. The (nonrecoding) binary method requires 6 point doublings plus 5 point additions in order to compute $119P$. In the recoding binary method, we first obtain a sparse signed-digit representation of 119. It is easy to verify the following:

$$\text{Exponent: } 119 = 01110111,$$
$$\text{Recoded Exponent: } 119 = 1000\bar{1}00\bar{1}.$$

The recoding binary method then computes $119P$ as follows:

f_i	Step 3	Steps 4-8
1	P	P
0	$2(P) = 2P$	$2P$
0	$2(2P) = 4P$	$4P$
0	$2(4P) = 8P$	$8P$
$\bar{1}$	$2(8P) = 16P$	$16P - P = 15P$
0	$2(15P) = 30P$	$30P$
0	$2(30P) = 60P$	$60P$
$\bar{1}$	$2(60P) = 120P$	$120P - P = 119P$

Table 4.3. A Toy Example of the Recoding Algorithm

The number of point doublings plus additions is equal to $7 + 2 = 9$ which is 2 less group operations than that of the binary method. The number of

point doubling operations required by the recoding binary method can be at most 1 more than that of the binary method. The number of subsequent point additions, on the other hand, can be significantly less. This is simply equal to the number of nonzero digits of the recoded exponent. Thus, the number of point addition operations can be reduced if we obtain a sparse signed-digit representation of the scalar k.

4.6.3 ω-NAF Representation

Algorithm 4.5 ω-NAF Expansion Algorithm

Require: A positive integer k.
Ensure: $U = \omega NAF(k)$
  for $\{i = 0; k > 0; i++\}$ **do**
    if k is odd **then**
      $U_i = k \bmod 2^w$
      $k = k - U_i$;
    else
      $U_i = 0$;
    end if
    $k = k/2$;
  end for
  Return(U);

The recoding binary algorithm can be generalized for designing algorithms even more efficient at the price of using memory for storing pre-computed results. The basic *window method* ω with $\omega > 1$ expand any positive integer k using a Non-Adjacent Form (NAF) of width ω expressed as,

$$k = \sum_{i=0}^{l-1} u_i 2^i$$

Where,

- Each coefficient u_i different than zero is odd and with magnitude less than 2^{w-1};
- Given two consecutive coefficients u_i, at least one of them is nonzero;
- When using $\omega = 2$ we have the recoding binary algorithm explained above.

We write the ωNAF as,

$$\omega NAF(k) = \{u_{l-1}, ...u_0\}.$$

Algorithm 4.5 generates an ωNAF expansion of a positive scalar k. Every time that k is odd, the ω most significant bits are scanned in order to determine

the corresponding congruence class (mod 2^ω) for k. The congruence class U_i is then subtracted from k, making the new coefficient $k - U_i$ divisible by 2^w. This will guarantee a run of $w - 1$ zero coefficients in the next iterations.

In average, the Hamming weight of a ωNAF expansion is $(w+1)^{-1}$. This will directly impact the performance of the scalar multiplication algorithm because of a saving on the point additions required for computing the scalar multiplication. That saving is obtained at the price of storing multiples of the base elliptic point. Notice, however, that the total number of point doublings remains the same. Table 4.4 presents the main characteristics of the binary, recoded binary an ωNAF expansions of the scalar k, respectively.

Table 4.4. Comparing Different Representations of the Scalar k

Point Representation	Length	# PA	# PD	Pre-computation
Binary	m	$\frac{m}{2}$	m	—
recoded binary	m	$\frac{m}{3}$	$m+1$	—
ωNAF	m	$\frac{m}{w+1}$	$m+1$	Table of $2^{w-1}-1$ m-bit multiples.

4.7 Conclusions

In this Chapter we briefly reviewed some of the most important mathematical concepts useful for understanding cryptographic algorithms. We explained the most relevant definitions and theorems of the elementary theory of numbers relevant to the subject of cryptography. Moreover, we defined the concept of finite fields and related arithmetic operations. We gave a brief introduction to elliptic curve cryptography, explaining the mathematical concepts of elliptic curve group, group order, group law and point representation among others.

These concepts will be useful for understanding the material contained in the Chapters to come.