A Survey of Elliptic Curve Cryptosystems, Part I: Introductory

San C. Vo
NASA Advanced Supercomputing (NAS) Division – Research Branch (INR)
Information Sciences & Technology Directorate
NASA Ames Research Center, Moffett Field, CA 94043

Introduction

The theory of elliptic curves is a classical topic in many branches of algebra and number theory, but recently it is receiving more attention in cryptography. An elliptic curve is a two-dimensional (planar) curve defined by an equation involving a cubic power of coordinate x and a square power of coordinate y. One class of these curves is elliptic curves over finite fields, also called Galois fields. These elliptic curves are finite groups with special structures, which can play naturally, and even more flexibly, the roles of the modulus groups in the discrete logarithm problems.

Elliptic curves have been used actively in designing many mathematical, computational and cryptographic algorithms, such as integer factoring, primality proving, public key cryptosystems and pseudo-random number generators, etc. Essentially, elliptic curve cryptosystems promise a better future for cryptography: more security against powerful attacks in the era of computing capability.

Many research papers in Elliptic Curve Cryptography (ECC) have been published by researchers all over the world. However, the idea of using elliptic curves in cryptography is still considered a difficult concept and is neither widely accepted nor understood by typical technical people. The problem may stem from the fact that there is a large gap between the theoretical mathematics of elliptic curves and the applications of elliptic curves in cryptography.

A large amount of ECC literature was collected and organized in the development of this survey on ECC. Part I (Introductory) of this survey gives a modest overview of how elliptic curves have been applied to public key cryptography. The objective is to introduce a bridge between the mathematical facts of elliptic curves and its application for cryptography. The document attempts to provide clear, intuitive and elementary explanations to guide a typical technical reader into the world of elliptic curve cryptography. However, basic knowledge of cryptography and abstract algebra, including group theory and number theory, would be helpful for readers in several technical areas. Part II of this survey, that will be developed, intends to focus more on practical implementations.

The materials cover elliptic curves and their basic mathematical rules, the Elliptic Curve Discrete Logarithm Problem (ECDLP) and many typical attacks on ECDLP-based cryptosystems. Also included are descriptions of elliptic curve public key cryptosystems or schemes (encryption/decryption, digital signature, key agreement and key transport schemes). The document concludes with discussions of elliptic curve implementations, the security and advantages of ECC.

It is hoped that this survey could provide readers good initial background on the path into the new and exciting area of elliptic curve cryptography, that is attracting more attention from cryptographers, computer scientists and researchers all over the world.

Contents

Notations & Acronyms

These are two tables of basic notations and acronyms that will be frequently used in the document.

Notation	Meaning	Notation	Meaning
F_q or $GF(q)$	Finite field of order q	$\Re(x + iy) =$	Real part of a complex

	(consisting of q elements)	$\Re(z) = x$	number $z = x + iy$
F_p or $GF(p)$	Prime finite field of order p (consisting of p elements)	$\Im(x + iy) = \Im(z) = y$	Imaginary part of a complex number $z = x + iy$
$F_q^* = F_q \setminus \{0\}$ $GF(q)^* = GF(q) \setminus \{0\}$	Multiplicative subgroup of the finite field F_q or $GF(q)$	$a \cdot P$, $(a \cdot P)$, aP or (aP)	Scalar point multiplication of an elliptic curve point P with a scalar a
F_{2^m} or $GF(2^m)$	Binary finite field of order 2^m (consisting of 2^m elements)	$a \mid b$	a divides b (or b is divisible by a)
$E(F_q)$ or $E(GF(q))$	Elliptic curve E over finite field F_q or $GF(q)$	$a \nmid b$	a does not divide b (or b is not divisible by a)
$\#E$ or $\#E(F_q)$ or $\#E(GF(q))$	Order of an elliptic curve E over finite field F_q or $GF(q)$	$Tr(\cdot)$	Trace function
$\#(F_q)$ or $\#(GF(q))$	Order of a finite field F_q or $GF(q)$	$\boldsymbol{Z}$	Ring of integers
$\tilde{E}$ or E'	Twisted curve of E	$\boldsymbol{Q}$	Field of rational numbers
$\langle P \rangle$ or $\langle g \rangle$	Group generated by an element P or g	$\boldsymbol{R}$	Field of real numbers
$P = (x_P, y_P)$ $P = (P_x, P_y)$	An elliptic curve point represented with its x- and y-coordinates	$[a, b]$	Closed interval consisting of all real values x, such that $a \le x \le b$
Δ	Discriminant	mod n	Modulo n
O	The point at infinity of an elliptic curve	$a \cdot b$ or ab	Scalar multiplication of two numbers
$\left(\frac{\cdot}{p}\right)$	Legendre symbol (modulo a prime number p)	lcm (a, b)	Least common multiplier of a and b
$\left(\frac{\cdot}{b}\right)$	Jacobi symbol (b is composite number)	gcd (a, b)	Greatest common divisor of a and b
Z_N, where $N = pq$	Ring of integers modulo N, where N is composite	$\lceil a \rceil$ $\lfloor a \rfloor$	The smallest integer $\ge a$ The biggest integer $\le a$

Table *i*. Notations

Acronym	Meaning	Acronym	Meaning
ABC	Anomalous binary curve or binary anomalous curve	MIPS	Million instructions per second
AES	Advanced encryption standard	MOV	Menezes-Okamoto-Vanstone algorithm
CM	Complex Multiplication	MQV	Menezes-Qu-Vanstone key agreement scheme
DES	Data Encryption Standard	MY	MIPS year
DH	Diffie-Hellman algorithm	NAF	Non-adjacent form
DHP	Diffie-Hellman problem	OEF	Optimal extension field

DDHP	Decision Diffie-Hellman problem	ONB	Optimal normal basis
DLP	Discrete logarithm problem	PDA	Personal Digital Assistant
DPA	Differential Power Analysis	PKI	Public key infrastructure
ECC	Elliptic curve cryptosystem	RSA	Rivest-Shamir-Adleman cryptosystem
ECDDHP	Elliptic curve Decision Diffie-Hellman problem	SHA-1	Standard – Secure hash algorithm, (revision 1)
ECDHP	Elliptic curve Diffie-Hellman problem	SKE	Symmetric key encryption
ECDLP	Elliptic curve discrete logarithm problem	SPA	Simple Power Analysis
ECDSA	Elliptic curve digital signature algorithm	SSA	Smart-Satoh-Araki attack
IFP	Integer factoring problem	VLSI	Very Large Scale Integration
KMOV	Koyama-Maurer-Okamoto-Vanstone cryptosystem		

Table *ii*. Technical Acronyms

Chapter 1 – **Elliptic Curves over Finite Fields**

The first chapter will introduce basic information about elliptic curves in order to build a foundation of the subject: definitions of the elliptic curves and finite fields, group structure of the elliptic curves and many other basic properties which are employed in cryptography.

Since 1985, two mathematicians Miller and Koblitz have been considered the co-founders of elliptic curve cryptography. It is a new branch in cryptography that uses an old, interesting and difficult topic in mathematics or, particularly, algebra: elliptic curves over finite fields. This has both fortunate and "unfortunate" consequences for elliptic curve cryptography. It is fortunate because ECC is based on a strong fundamental mathematical background. This makes the solution of the Elliptic Curve Discrete Logarithm Problem still infeasible; hence it still serves as the security core of elliptic curve cryptosystems. The "unfortunate" aspect is that the background of ECC is too complicated to be explained elementarily. The theory of RSA (Rivest-Shamir-Adleman) cryptosystems, which are based on the Integer Factoring Problem, is fortunately rather easy to discuss using high-school mathematics. The cryptosystems, which are based on the Discrete Logarithm Problem defined on a finite field, require only elementary number theory knowledge about modular multiplications and additions. To understand the theory of ECC, the reader must study elliptic curves and get familiar with basic mathematical concepts related to elliptic curves.

1.A. Finite fields

1.A.1. Basic facts

Here, we state briefly, without proof, basic facts on finite fields, necessary for further discussion on the subject. For more details, please refer to books on the theory of finite fields.

Finite fields, also called Galois fields, are fields consisting of a finite number of elements.

A finite field has p^m elements where p is a prime number and m is a natural number. We call p^m the order of the finite field. There is no finite field of order n, if n is not a positive power of a prime number.

For any prime p and positive integer m, there always exists a Galois field of order p^m.

Two finite fields of the same number of elements p^m are isomorphic, or roughly speaking, they are the same finite field. We then call it *the* Galois field (or finite field) of order p^m, which is denoted by either $GF(p^m)$ or F_{p^m}.

We call p the characteristic of the finite field $GF(p^m)$. In general, the characteristic of a field F is the smallest positive integer p such that $p \cdot 1 = \underbrace{1+\cdots+1}_{p \text{ terms}} = 0$, where 1 is the multiplicative identity of the field. We write $\text{char}(F) = p$. Then the arithmetic operations over a finite field are reduced simply to multiplications and additions modulo p.

It can be shown that p must be a prime number, because otherwise we can find a prime factor p_1 of p, (hence $p_1 < p$) such that $p_1 \cdot 1 = 0$. The contradiction on the minimum of prime p proves the claim.

A cyclic group G is a group that can be generated from any one of its elements. For multiplicative cyclic group G (whose operation is a multiplication $\cdot$) and any given element g in G, there is an element a in G and a positive number k such that: $a^k = \underbrace{a \cdot a \cdot \ldots \cdot a}_{k \text{ terms}} = g$. For an additive cyclic group (whose operation is an addition +) and any given element g in G, there is an element a in G and a positive number k such that: $ka = \underbrace{a + a + \cdots + a}_{k \text{ terms}} = g$.

The multiplicative subgroup of a finite field F_q, written as $F_q^* = F_q \backslash \{0\}$ is consisting of non-zero elements, or invertible elements of a finite field F_q. This group is cyclic of order $(q - 1)$. Particularly, we have: $a^{q-1} = 1$, $\forall a \in F_q^*$.

1.A.2. Prime and binary finite fields

Let F_q be the Galois finite field of q elements, where $q = p^m$ for some prime p and positive integer m.

When $m = 1$, we usually denote the finite field F_p by Z/pZ or Z_p (if there will be no confusion with the p-adic field). The arithmetic operations on F_p are the usual addition and multiplication modulo p.

When $m \geq 2$, then we have: $F_{p^m} \not\equiv Z_{p^m}$. In fact, Z_{p^m} is only a ring, and it is even not a field, let alone Galois field, since, for instance, any element that is a multiple of p is equal to 0, not 1; hence it has no inverse.

From now on, we will consider only elliptic curves $E(F_q)$ defined over finite field F_q, where $q = p^m$ and $\text{char}(F_q) = p$. Particularly, there are two cases:

If p is a prime number > 3, then $q = p$. In this case, the finite field is F_p and its characteristic char(F_p) = p > 3. Particularly, F_p = {0, 1, 2,…, (p – 1)} whose field operations are addition and multiplication modulo p. It is called a prime finite field.

If $p = 2$, then $q = 2^m$ for some positive integer m. The finite field is $GF(2^m)$ or F_{2^m} and its characteristic is char ($GF(2^m)$) = 2. It is called a binary finite field.

The simplest case of a finite field is when $m = 1$. The finite field $F_2 = \{0,1\}$ has the following operations:

$0 + 0 = 1 + 1 = 0$ $\quad$ $0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0$

$1 + 0 = 0 + 1 = 1$ $\quad$ $1 \cdot 1 = 1$.

We consider a less obvious case $m = 2$: the finite field $GF(2^2)$. Let $\alpha^3 = 1$ and $\alpha \neq 1$. The elements of $GF(2^2)$ are 0, 1, α and $\alpha^2 = \alpha + 1$. Using the standard trinomial basis, we can write: $GF(2^2) = \{(00) = 0, (01) = 1, (10) = \alpha, (11) = \alpha^2\}$. Simple operations on $GF(2^2)$ are:

$\alpha + \alpha = \alpha^2 + \alpha^2 = 1 + 1 = 0$ $\quad$ $\alpha + \alpha^2 = 1$ $\quad$ $1 + \alpha^2 = \alpha$

$\alpha \cdot \alpha = \alpha^2$ $\quad$ $\alpha \cdot \alpha^2 = 1$ $\quad$ $\alpha^2 \cdot \alpha^2 = \alpha$

1.B. The curve and the group

Let E be an elliptic curve over a finite field F_q whose equation is given as follows.

Over prime finite field F_p where prime $p \neq 2, 3$, we use elliptic curves of equation

$$y^2 = x^3 + ax + b, \text{ where } \Delta = -16(4a^3 + 27b^2) \neq 0.$$

Over binary finite field $GF(2^m)$, we use non-supersingular elliptic curves of equation: $y^2 + xy = x^3 + ax^2 + b$, where $\Delta = b \neq 0$.

We will also discuss supersingular elliptic curves over binary finite field $GF(2^m)$ whose equations are of the form:

$$y^2 + cy = x^3 + ax + b, \text{ where } \Delta = c^4 \neq 0.$$

All the points of $E(F_q)$, including the point at infinity, (which is denoted conveniently as O), form an abelian (or commutative) group whose identity element is O and the group law is addition +, which is defined as follows. We will then consider E as both a curve and a group simultaneously. This group of elliptic curve points is the group on which an elliptic curve cryptosystem will be defined. It is similar to the case of Diffie-Hellman cryptosystem, (Diffie & Hellman [DH76] or more generally, any cryptosystem based on the Discrete Logarithm Problem, defined on a group of elements of a prime finite field.

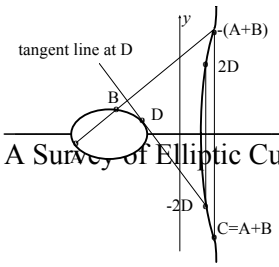
Additive inverse of a point *P*

First, we have –O = O. For any point $P = (x, y) \neq$ O on E, the additive inverse point ($-P$) of point P is defined as in the table 1.1.

Equation of elliptic curve E over a finite field F_q	$-P = -(x, y)$
$y^2 = x^3 + ax + b$ (over F_p where prime $p \neq 2, 3$)	$-P = (x, -y)$
$y^2 + xy = x^3 + ax^2 + b$ (non-supersingular elliptic curve over $GF(2^m)$)	$-P = (x, y + x)$
$y^2 + cy = x^3 + ax + b$ (supersingular elliptic curve over $GF(2^m)$)	$-P = (x, y + c)$

Point addition rules

For any point $P \neq$ O and $Q \neq$ O on E, we have $P + Q = Q + P = R$, where the inverse point ($-R$) is the intersection point of the elliptic curve E with the line going through P and Q if $P \neq$ Q or with



the tangent line to the elliptic curve at the point P if P = Q. (There is exactly one such point of intersection ($-R$), since the intersection of a straight line and a cubic curve will give at most 3 points.) The addition rules, which are also called chord-and-tangent laws, will be best illustrated with the field of real numbers $\boldsymbol{R}$ as in the graph in Figure 1.1.
Figure 1.1. Point addition rules (or chord-and-tangent rules) for an elliptic curve

However, for finite fields, an elliptic curve is not a continuous curve, but it is a collection of scattered points and the point at infinity O (which is not drawn). Refer to Figure 1.2 and 1.3.

For example, let us consider the elliptic curve E: $y^2 = x^3 + x + 6$ over the finite field F_{11}, which has order $\#E(F_{11}) = 13$.
$E = \langle P \rangle = \{(2,7), (5,2), (8,3), (10,2), (3,6), (7,9), (7,2), (3,5), (10,9), (8,8), (5,9), (2,4), O\}$.

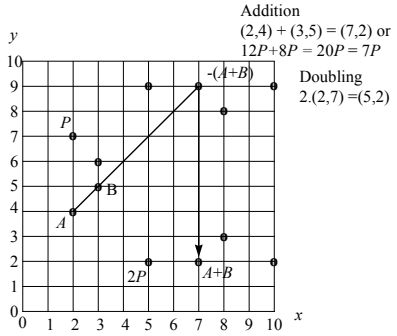
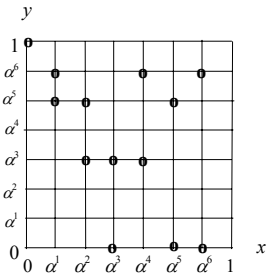


Figure 1.2.
The graph of an elliptic curve $E((GF_{11}))$: $y^2 = x^3 + x + 6$ includes 12 points and the point at infinity O.
The number of points is $\#E((GF_{11})) = 13$.
The point addition rules over finite field GF_{11}.

Figure 1.3.
The graph of an elliptic curve $E(GF(2^3))$: $y^2 = x^3 + x + 1$ has 13 points and the point at infinity O.
The number of points is $\#E(GF(2^3)) = 14$.



We have, trivially, $P + (-P) = O$. One can imagine that the third intersection point of the elliptic curve and the line going through 2 points P and $(-P)$ is at infinity. Hence O is called the point at infinity.

The explicit formulae for the addition of two non-identity points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ in all three cases discussed above are given in table 1.2. There is a basic property of cubic equations that was used in deriving those formulae.

If x_1, x_2 and x_3 are three roots of a cubic equation $X^3 + aX^2 + bX + c = 0$, then

$$x_1 + x_2 + x_3 = -a.$$

Point doubling formula

When $P = Q$, the addition formula is called the formula for doubling a point P. This is the basic arithmetic for scalar point multiplication that will be used the most in implementations of elliptic curves.

Scalar multiplication of a point (or Scalar point mutiplication)

Given a point P on an elliptic curve E and an integer k, the scalar point multiplication of P by k is the point $k \cdot P$ that is computed by the following formula:

$$k \cdot P = \underbrace{P + P + \cdots + P}_{k \text{ terms}} \text{ if } k > 0, \text{ and } k \cdot P = (-k) \cdot (-P), \text{ if } k < 0.$$

The notation $k \cdot P$ is usually replaced by other equally popular notations: kP, (kP) or $(k \cdot P)$ where the context is clear.

Obviously, we have: $0 \cdot P = P + (-P) = \mathrm{O}$.

Equation of elliptic curve E over a finite field F_q	$R = (x_3, y_3) = (x_1, y_1) + (x_2, y_2)$ where
Over F_p where prime $p \neq 2, 3$ $y^2 = x^3 + ax + b$, $\Delta = -16(4a^3 + 27b^2) \neq 0$ $P = (x_1, y_1)$ $Q = (x_2, y_2)$	$x_3 = \lambda^2 - x_1 - x_2$ $y_3 = \lambda (x_1 - x_3) - y_1$ where $\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq \pm Q \\ \dfrac{3x_1^2 + a}{2y_1} & \text{if } P = Q. \end{cases}$ If $y_1 = 0$, then $P = (x_1, 0) = -P$. Hence $2.P = \mathrm{O}$.
Non-supersingular elliptic curve over $GF(2^m)$ $y^2 + xy = x^3 + ax^2 + b$, $b \neq 0$ $P = (x_1, y_1)$ $Q = (x_2, y_2)$ Let $\kappa = \dfrac{y_1 + y_2}{x_1 + x_2}$ and $\mu = x_1 + \dfrac{y_1}{x_1}$.	$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + \dfrac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a \\ \qquad = \kappa^2 + \kappa + x_1 + x_2 + a & \text{if } P \neq \pm Q \\ x_1^2 + \dfrac{b}{x_1^2} = \mu^2 + \mu + a & \text{if } P = Q, \end{cases}$ $y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1 \\ \qquad = \kappa.(x_1 + x_3) + x_3 + y_1 & \text{if } P \neq \pm Q \\ x_1^2 + \left(x_1 + \dfrac{y_1}{x_1}\right).x_3 + x_3 \\ \qquad = x_1^2 + (\mu + 1).x_3 & \text{if } P = Q. \end{cases}$ If $x_1 = 0$, then $P = (0, \sqrt{b}) = -P$. Hence $2P = \mathrm{O}$.
Supersingular elliptic curve over $GF(2^m)$ $y^2 + cy = x^3 + ax + b$, $c \neq 0$ $P = (x_1, y_1)$ $Q = (x_2, y_2)$ Let $\kappa = \dfrac{y_1 + y_2}{x_1 + x_2}$ and $\eta = \dfrac{x_1^2 + a}{c}$	$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + x_1 + x_2 \\ \qquad = \kappa^2 + x_1 + x_2 & \text{if } P \neq \pm Q \\ \dfrac{x_1^4 + a^2}{c^2} = \eta^2 & \text{if } P = Q, \end{cases}$ $y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + y_1 + c \\ \qquad = \kappa.(x_1 + x_3) + y_1 + c & \text{if } P \neq \pm Q \\ \left(\dfrac{x_1^2 + a}{c}\right)(x_1 + x_3) + y_1 + c \\ \qquad = \eta.(x_1 + x_3) + y_1 + c & \text{if } P = Q. \end{cases}$

Table 1.2. Point addition formulae for elliptic curves over finite fields

The order n of a point P is the smallest positive integer such that $n \cdot P = O$.

In fact, the graph of an elliptic curve over a finite field consists of a finite number of points (together with the point at infinity). It is not a continuous curve as in the case where an elliptic curve is defined over real numbers. But we still have a similar property: a line going through any two points will pass through one and only one other point.

1.C. Order of the group over finite fields

1.C.1. Hasse's theorem

The order of the group $E(F_q)$ is $\#E(F_q) = q + 1 - t$, where $| t | \le 2\, q^{1/2}$.

By this theorem, the order of an elliptic curve is roughly about the order of the finite field. For any element $a \in F_q$, if it is the x-coordinate of a point P in $E(F_q)$, then it is also the x-coordinate of the point $(-P)$. Hence, the probability for $a \in F_q$ to be the x-coordinate of a point in $E(F_q)$ is roughly equal to ½.

This theorem is commonly called the Riemann Hypothesis for elliptic curves over a finite field. It was proved, for many of its cases, by Artin in his Ph.D. thesis, for elliptic curves by Hasse, and for curves of higher genus. Note that this is not the same as the Riemann Hypothesis for the Riemann zeta function.

Property: In general, any value $| t | \le 2p^{1/2}$ can occur if gcd $(t, p) = 1$, for any characteristic p. Particularly:

When $q = p$, a prime, every possible value of t (i.e., $| t | \le 2p^{1/2}$) can be attained by some elliptic curve. When $| t | \le p^{1/2}$, the elliptic curves are roughly equally distributed.

When q is even (or $p = 2$), every odd value of t such that $| t | \le 2q^{1/2}$ can be attained by some non-supersingular elliptic curve.

Waterhouse's lemma ([W69]): *For $q = p^m$, there exists an elliptic curve E over a finite field F_q such that the elliptic curve order $\#E(F_q) = q + 1 - t$, if and only if one of the following conditions holds:*

(*i*) $t \not\equiv 0 \pmod p$ *and* $t^2 \le 4q$.

(*ii*) *m is odd and one of the followings holds:*

(1) $t = 0$.

(2) $t^2 = pq = p^{m+1}$ *if* $p = 2$ *or* 3.

(*iii*) *m is even and one of the followings holds:*

(1) $t^2 = 4q$.

(2) $t^2 = q$ *and* $p \not\equiv 1 \pmod 3$.

(3) $t = 0$ *and* $p \not\equiv 1 \pmod 4$.

1.C.2. Formulae and algorithms on elliptic curve group orders

Hasse-Weil's theorem (Weil's conjecture, proved by Helmut Hasse in 1934.)

Let E be an elliptic curve over a finite field F_q. Then E is also an elliptic curve over an extension field $GF(q^k)$ of F_q. As a group, we have the inclusion relationship: $E(F_q) \subset E(GF(q^k))$. That is, t*he elliptic curve order $\#E(F_q)$ must divide the elliptic curve order $\# E(GF(q^k))$. Moreover, if $\#E(F_q) = q + 1 - t$, then $\# E(GF(q^k)) = q^k + 1 - \alpha^k - \beta^k$, where α and β are complex numbers satisfying the equation:*

$$qT^2 - tT + 1 = (1 - \alpha T)\,(1 - \beta T).$$

This theorem helps to compute the order of an elliptic curve defined over a composite extension finite field from the order of the same elliptic curve over one of its

subfields. Explicitly, we have two relationships between elements α and β: $\alpha + \beta = t$ and $\alpha\beta = q$. Then the order $\#E(GF(q^k))$ can be computed via the sum $S_k = \alpha^k + \beta^k$ that is given by the following recursive formula (also called a Lucas sequence):

$S_k = (\alpha + \beta)S_{k-1} - \alpha\beta S_{k-2} = t\,S_{k-1} - q\,S_{k-2}$, for $k \geq 2$, where $S_0 = 2$ and $S_1 = \alpha + \beta = t$.

Direct formulae for the orders of elliptic curves over finite fields

When the finite field is of computationally small order, one still can use direct formulae to find the order of an elliptic curve. For each and every element $x \in F_q$, we will determine whether there is (are) 0, 1 or 2 corresponding values of y by the Legendre symbol (in prime finite field F_p) and the trace function (in binary finite field $GF(2^m)$). Summary of results is in the table 1.3.

Equation of elliptic curve E over a finite field F_q	Order of the elliptic curve $\#E(F_q)$
Over F_p, $p \neq 2, 3$ $y^2 = x^3 + ax + b$ $\Delta = -16(4a^3 + 27b^2) \neq 0$	$\#E = 1 + \sum_{x \in F_p} \left[\left(\frac{x^3 + ax + b}{p} \right) + 1 \right] = p + 1 \sum_{x \in F_p} \left(\frac{x^3 + ax + b}{p} \right)$, where $\left(\frac{*}{p}\right)$ denotes the Legendre symbol.
Non-supersingular elliptic curve over $GF(2^m)$ $y^2 + xy = x^3 + ax^2 + b$ $b \neq 0$	$\#E = 2 + \sum_{0 \neq x \in F_{2^m}} \left(1 + (-1)^{Tr(a + x + b.x^{-2})}\right)$ $= 2^m + 1 + (-1)^{Tr(a)} \sum_{0 \neq x \in GF(2^m)} (-1)^{Tr(x + bx^{-2})}$ When $x = 0$, we always have one solution: $y = b^{1/2} = b^{2^{m-1}}$. The order must be an even number.
Supersingular elliptic curve over $GF(2^m)$ $y^2 + cy = x^3 + ax + b$ $c \neq 0$	$\#E = 1 + \sum_{x \in GF(2^m)} \left(1 + (-1)^{Tr[c^{-2}(x^3 + ax + b)]}\right)$ $= 2^m + 1 + (-1)^{Tr(c^{-2}b)} \sum_{x \in GF(2^m)} (-1)^{Tr[c^{-2}(x^3 + ax)]}$ The order must be an odd number of possible values: $\#E = 2^m + 1$, $2^m + 1 \pm 2^{m/2}$, $2^m + 1 \pm 2^{(m+1)/2}$, or $2^m + 1 \pm 2^{(m+2)/2}$.

Table 1.3. Direct formulae for computing the orders of elliptic curves over finite fields

Other algorithms

Many particular elliptic curves over particular finite fields, whose the orders are easily computed or formulated, are implemented in cryptography for different purposes. For examples, the Koblitz curves or elliptic curves over a prime finite field F_p of the form

$E_p(a,0)$: $y^2 = x^3 + ax$, for $a \not\equiv 0 \pmod p$ or $E_p(0,b)$: $y^2 = x^3 + b$, for $b \not\equiv 0 \pmod p$.

For a general elliptic curve over larger finite field F_q, one should use Shanks' Baby-step-Giant-step algorithm (Buchmann & Müller [BM91]). The idea is to pick up a random point P on the elliptic curve and to compute an integer n such that: $q + 1 - 2q^{1/2} \leq n \leq q + 1 + 2q^{1/2}$ and $nP = O$. If we can find only one such number, then it is the order of the elliptic curve. If not, we find another point and continue. The groups, generated by all the points that we picked, will eventually have the order of the elliptic curve. Its running time is about $O(q^{1/4})$.

Shanks' Baby-step-Giant-step algorithm can fail when multiple values of n are available for every point P. Mestre [M86] showed that if Shanks' algorithm fails for an elliptic curve, then it will not fail on its twisted curve.

If an elliptic curve has Complex Multiplication properties, there are other efficient algorithms to count the points. Refer to Lenstra & Lenstra [LL90], Atkin & Morain [Mo91] and Lay & Zimmer [LZ94].

In 1985, Schoof's algorithm, which is of polynomial running time, was proposed and later has been improved both theoretically and practically to be used to compute the order of an elliptic curve over very large finite fields.

1.C.3. Supersingularity

An elliptic curve over a finite field F_q, where $q = p^m$, is supersingular if p divides t, where $\#E(F_q) = q + 1 - t$. If otherwise, it is a non-supersingular elliptic curve.

Lemma (*i*) *If char*$(F_q) = 2$ (*or* 3)*, the elliptic curve* $E(F_q)$ *is supersingular if and only if its j-invariant is equal to* 0.

(*ii*) *The elliptic curve* $E(F_q)$ *is supersingular if and only if either* $t^2 = 0, q, 2q, 3q$ or $4q$.

From the lemma, we observe that: *If q is even* (*i.e.,* $q = 2^m$)*, then E is non-supersingular, when* $\#E(GF(2^m))$ *is even or t is odd. Otherwise, if E is supersingular, when t is even, or the order* $\#E(GF(2^m))$ *is odd.*

Explicitly, the possible values of $\#E(GF(2^m))$ for supersingular elliptic curves are:

$$2^m + 1,\ 2^m + 1 \pm 2^{m/2},\ 2^m + 1 \pm 2^{(m+1)/2} \text{ and } 2^m + 1 \pm 2^{(m+2)/2}.$$

Each value can be attained provided it is an integer, of course.

If $q = p$*, a prime* > 3*, then the elliptic curve* $E(F_q)$ *E is supersingular if and only if its order* $\#E(F_p) = p + 1$.

Indeed, the simple reason is that $| t | \le 2p^{1/2} < p$. Hence the condition $p|t$ implies that $t = 0$.

The class of supersingular elliptic curves also is an interesting area of research in cryptography. The supersingular elliptic curves over either finite fields F_p or $GF(2^m)$ are vulnerable to the MOV attack (Menezes, Okamoto & Vanstone [MOV93]).

We now re-state Waterhouse's lemma in a different way, taking consideration of supersingularity [BS91].

Lemma: *There exists an elliptic curve E over a finite field* F_q *where* $q = p^m$*, such that* $\#E(F_q) = q + 1 - t$*, if and only if one of the following conditions holds:*

(*i*) *For the case of supersingular curves,*

m is even: $t = \pm 2q^{1/2} = \pm 2p^{m/2}$.

m is even and $p \not\equiv 1 \pmod 3$: $t = \pm q^{1/2} = \pm p^{m/2}$.

m is odd and $p = 2$ *or* 3: $t = \pm (pq)^{/12} = \pm p^{(m+1)/2}$.

m is odd or m is even and $p \not\equiv 1 \pmod 4$: $t = 0$.

(*ii*) *For the case of non-supersingular curves, two conditions* $| t | \le 2q^{1/2}$ *and* $\gcd(t, p) = 1$ *must hold.*

1.C.4. Structure of the group

The group $E(F_q)$ is either a cyclic group or a direct sum of two cyclic groups $Z_{n_1} \oplus Z_{n_2}$, where $n_2 | n_1$ and $n_2 | (q - 1)$. That is, we have $n_2 | \gcd(n_1, q - 1)$.

The elliptic curve has order $\# E(F_q) = n_1 n_2$.

This result together with the theory of abelian groups can help us determine the group structure. One effective technique in algebra is counting the number of points of particular order. There are many special cases to consider. In an easiest case, the order of a point to be considered is 2. In fact, points of order 2 are of the form $(x, 0)$ when $\text{char}(F_q) \neq 2, 3$. For non-supersingular elliptic curves when $\text{char}(F_q) = 2$, there is only one point of order 2. It is $(0, {}^{1/2})$.

Lemma: *If the order $\#E(F_q)$ is square-free, then $E(F_q)$ is a cyclic group.*

In other words, if any prime number s such that $s^2 | \#E(F_q)$ must satisfy the condition $s \nmid (q - 1)$, then the group $E(F_q)$ is cyclic.

1.C.5. Schoof's algorithm and improvements

This is a short historical summary of developments of the Schoof's algorithm since its original version in 1985. Many developed steps of the complete algorithm are described in more details in the cited references.

a. Schoof ([S85],[S87]) presented an algorithm for counting the number of points N on an elliptic curve. It originally applied in the case where $q = p^m$ and $p > 3$.

Schoof used the theory of division polynomials and Frobenius endomorphisms to determine the order N (modulo l) for a collection of small primes l. If the number of such primes is large enough, such that the product of such primes $\prod l > 4q^{1/2}$, then one can use the Chinese remainder theorem to determine the number N uniquely. Schoof's algorithm is a deterministic method. This algorithm has a running time of about $O(\ln^8 q)$ or $O(\ln^9 q)$.

Buchmann & Müller [BM91] gave experimental results using a combination of Schoof's algorithm and Shanks' Baby-step-Giant-step algorithm for the case $p > 3$. First, we compute the order $\#E(F_q)$ mod $(l_1 l_2 \ldots l_s)$ for a few small primes $l_1, l_2, \ldots, l_s$, using Schoof's algorithm. This helps to reduce the table size in the Baby-step-Giant-step algorithm by a factor of the inverse product $(l_1 l_2 \ldots l_s)^{-1}$.

Menezes, Vanstone & Zuccherato [MVZ93] discussed Schoof's algorithm implemented for finite fields of characteristic 2.

b. Since Schoof's algorithm's running time is too slow, it was not practical in applied cryptography. Later, improvements in both theory and implementation were made by Elkies and Atkin in their unpublished manuscripts from 1986 to 1992. Atkin and Elkies used the properties of modular polynomials to get the possible values of N modulo l. This method introduced new concepts: Atkin primes and Elkies primes. It involves the isogeny between two l-isogenous elliptic curves. Atkin and Elkies also proposed using more modular equations and modular forms to improve the implementation.

The improvements have provided more efficient implementation of Schoof's algorithm over finite fields of any large characteristic. The improved Schoof's algorithm is then called Schoof-Elkies-Atkin (SEA) algorithm.

Implementations of this algorithm were presented in a few works, such as Morain [Mo95] and Joux & Lercier [JL]. Couveignes & Morain [CM94] showed an improvement in case of powers of small Elkies primes by using a structure called an "isogeny cycle." Lehmann, Maurer, Müller & Shoup [LMMS94] presented a variant of Atkin's method and its implementation. More details were also organized in Schoof [S95]. The author also discussed Mestre's method and Cornacchia's algorithm.

Mestre's method is to simplify to Shanks' Baby-step-Giant-step algorithm in the special cases. This method is practical for finite fields that are not too large.

Cornacchia's algorithm uses the natural lattice structure of the endomorphism ring of the elliptic curve and then it is very effective if the ring is known. It is the basis for the primality test by Atkin & Morain [Mo91].

c. The problem remains for the case $p = 2$, or in fact, for small characteristics less then l, the prime degree of isogenies. Couveignes [C94] proposed a method to work for any p, hence, helped to solve the case $p = 2$, using the formal group associated to an elliptic curve. It is an important theoretical breakthrough that speeds up the computations significantly. The first successful implementation of this algorithm was presented in Lercier & Morain ([LM95], [LM95a]). Its running time is of $O(l^3)$ elementary field operations and need a memory storage of $O(l^2)$. The main cost for a counting algorithm in this case is the computation of isogenies of prime degree l between two isogenous elliptic curves.

d. Later, Lercier [L96] proposed a heuristic algorithm for computing the isogenies, for the case of characteristic 2, to replace that method in Couveignes' algorithm since this is too slow and requires too much memory. Lercier's method works on the elliptic curve itself instead of on its formal group. Its running time is also $O(l^3)$ field operations, but it is faster by a significant constant factor in practice and conceptually simpler. The memory space required is reduced to only $O(l)$. It is not known whether this method can be generalized to the case of any other characteristic.

In turn, Couveignes proposed a general algorithm (method II) using elementary Galois properties of the p-torsion points, without using the formal group. It works for any characteristic. It was based on the ideas in Lercier & Morain's work above.

This algorithm allows the use of fast multiplication for polynomials to achieve its running time at $O(l^{2+\varepsilon})$ field operations. This method seems to reduce significantly the burden of implementation of a point counting algorithm. The memory required is also $O(l)$.

e. References on current developments of the algorithm are discussed in Couveignes [C96], Couveignes, Dewaghe & Morain [CDM96], Müller & Paulus[MP97], Dewaghe [D98], Lercier ([L97],[L97a]), Elkies [E98], Galbraith [G99] and later works. Some surveys are helpful to understand the general ideas: Buchmann, Müller & Shoup [BMS95] and Lercier & Morain ([LM95a],[LM97]).

f. Satoh [Sa00] proposed a completely different algorithm (from Schoof-Elkies-Atkin algorithm) with running time $O(\log^{3+a} q)$ or $O(\log^5 q)$ for small fixed characteristic $p \geq 5$ and suggested that algorithm can be applied for the case of characteristic 2 and 3. Satoh's method is based on the canonical p-adic lift of an ordinary elliptic curve.

Fouquet, Gaudry & Harley [FGH00] extended Satoh's method to the case of characteristic 2 and 3. The authors also showed in [FGH01] an implementation of the Satoh-FGH algorithm and an early-abort strategy based on the Schoof-Elkies-Atkin (SEA) algorithm to find secure random elliptic curves in finite fields of characteristic 2 in a faster time than previous methods.

Chapter 2 – **Elliptic Curve Cryptosystems**

We will discuss basic cryptosystems using elliptic curves, such as encryption/decryption, digital signatures and key agreement, etc. For each scheme, we provide brief discussions and analyses on their security against some known attacks that are already published. We also mention RSA-type elliptic curve cryptosystems that have interesting applications.

2.A. Introduction to elliptic curve cryptosystems

2.A.1. The discrete logarithm problem

The discrete logarithm problem can be defined over any abelian group. For simplicity, we recall the definition of DLP over finite cyclic group.

Let G be a finite cyclic group of order n and generated by g, i.e., $G = \langle g \rangle$. Given a point $y \in G$, find an integer m, $0 \le m \le n - 1$, such that $y = g^m$.

We call $m = \log_g y$ the discrete logarithm of y to the base element g.

The DLP can be solved in a sub-exponential running time.

In cryptography, one usually takes group G as the cyclic (multiplicative) subgroup of a modulo group Z_p where p is a prime number. Then

$$G = Z_p^* = Z_p \setminus \{0\} \text{ and its order is } |G| = |Z_p^*| = p - 1.$$

Let g be a generator of G. Then given $y \in G = \langle g \rangle = Z_p^*$, the DLP is to find an integer m, $0 \le m \le p - 1$, such that $y = g^m \pmod p$.

More generally, group G can be the multiplicative group in a finite field $GF(p^m)$ Then its order is $n = |G| = |GF(p^m)^*| = p^m - 1$. In fact, for cryptographic security reasons, one prefers to work on a cyclic subgroup of G whose order is a prime number that divides the number $(p^m - 1)$.

A variety of groups are used in this problem instead of the modulo groups. Now we will consider using the group of points on an elliptic curve over finite fields.

2.A.2. The cryptographic problems on elliptic curves

a. The elliptic curve discrete logarithm problem

Given a point P of order n in an elliptic curve E over a finite field F_q and a point Q in the subgroup of E generated by P, denoted by $\langle P \rangle$, the ECDLP is to find an integer m, where $0 \le m \le n - 1$, such that $Q = m \cdot P$.

Another way to describe the problem is:

Given a point P of order n in an elliptic curve E over a finite field F_q and a point Q in E, the ECDLP is to find an integer m, $0 \le m \le n - 1$, such that $Q = m \cdot P$ if such a number exists.

Point P is called the base point in this problem. We call $m = \log_P Q$ the elliptic curve discrete logarithm of Q to the base point P.

The ECDLP is believed to be unsolvable in sub-exponential time, while there are already algorithms to solve the DLP in sub-exponential time.

The following lemma from group theory tells us whether a solution for an ECDLP exists.

Lemma: *Let N be the order of the elliptic curve E and n be the order of the subgroup $\langle P \rangle$ (generated by point P). Let $l = N/n$ be the cofactor, or the index, of subgroup $\langle P \rangle$. If* $\gcd(n, l) = 1$*, then a point Q in the elliptic curve E is in subgroup $\langle P \rangle$ if and only if $n \cdot Q =$* O.

The condition gcd(n, l) = 1 is easily satisfied, since in practice, n should be a prime number.

b. Elliptic curve Diffie-Hellman problem

Elliptic curve Diffie-Hellman problem (ECDHP) ([DH76]): *Given a point P of order n in an elliptic curve E over a finite field F_q and two points (k·P) and (l·P),the ECDHP is to find the point (kl·P).*

This problem is also used in the elliptic curve Diffie-Hellman key exchange algorithm. Boneh & Lipton [BL96] proved that: If the ECDLP cannot be solved in sub-exponential time, then neither can the ECDHP.

Elliptic curve decision Diffie-Hellman problem (ECDDHP) (Boneh [B98]): *Given a point P of order n in an elliptic curve E over a finite field F_q and three points (k·P), (l·P) and (m·P), the ECDDHP is to decide whether m = kl (modulo the order of point P).*

The ECDDHP is not harder than the ECDHP. Boneh & Venkatesan [BV96] and Boneh & Shparlinski [BS01] discussed many issues on security of the ECDHP and related schemes.

2.A.3. Approaches in elliptic curve cryptosystems

In 1985, Koblitz and Miller independently introduced the use of the group of points of an elliptic curve over a finite field in cryptosystems based on the elliptic curve discrete logarithm problem. (Koblitz' work [K87] was not published until two years later than Miller's [Mi85]).

They are called elliptic curve cryptosystems. In the current cryptography literature, there are basically three approaches.

(*i*) Diffie-Hellman/DSA-type elliptic curve cryptosystems

Many proposed elliptic curve cryptosystems of this type are mentioned, such as ElGamal, Menezes-Vanstone, Massey-Omura, Schnorr, Nyberg-Rueppel schemes, MQV schemes. They are also used in many Standards such as ANSI X9.62 & ANSI X9.63 and IEEE P1363.

(*ii*) RSA-type elliptic curve cryptosystems: KMOV, Demytko schemes and other schemes:Koyama's "K scheme," Koyama & Kuwakado, Meyer-Müller, Chua-Ling, Fujioka-Fujisaki-Okamoto and McCurley schemes.

(*iii*) Other elliptic curve cryptosystems: using supersingular elliptic curves or Koblitz curves.

A Koblitz curve over $GF(2^m)$ is a non-supersingular elliptic curve whose defining function has coefficients in F_2. In literature, they are also referred as anomalous binary curves (ABC's) or binary anomalous curves. These curves must have the form E_a: $y^2 + xy = x^3 + ax^2 + 1$, where a = 0 or 1. These curves are non-supersingular elliptic curves (in order to resist MOV attack) and not vulnerable to the SSA attack. They are easy to create and implementbecause of Complex Mulitplication properties.

We will present the outline of practical approaches of elliptic curve cryptosystems that are proposed in cryptography literature. They are grouped into: message encryption/decryption schemes, digital signature schemes with and without message recovery, authentication schemes, key exchange and key agreement schemes, RSA-type elliptic curve cryptosystems, and elliptic curve digital signature schemes over a ring Z_N.

Each scheme or cryptosystem will be presented in its basic algorithms. Simple analyses on security that should be widely known in literature and a few typical features will be also pointed out where possible.

2.A.4. Public key and private key generation

Let E be a non-supersingular elliptic curve defined over the finite field F_q, where either $q = 2^m$, for some positive integer m or q is a prime greater than 3. Choose P as a base point of a big prime order n in $E(F_q)$. We will work on the group generated by the point P: $\langle P \rangle = \{O, P, 2P, \ldots, (n-1)P\}$.

Key generation: The signer will select a random integer d in the interval $[2, n-2]$, and compute the point $Q = d \cdot P$. The private key is d and public key is Q or, in fact, a pair of points P and Q on the elliptic curve.

2.B. Message encryption/decryption schemes

Unless stated otherwise, the elliptic curve E is a non-supersingular curve defined over a (Galois) finite field F_q, where q is a prime $p > 3$, or $q = 2^m$, where m is a positive integer. Simple analyses on security and specification features are occasionally discussed.

2.B.1. Elliptic curve analogue of the ElGamal cryptosystem ([E85])

A point P of order n on an elliptic curve E is fixed and publicly known. User A chooses a random number d, keeps it secretly, and publishes the key $Q = d \cdot P$. To send a message m to user A, user B first embeds m to a point $M = (m_x, m_y)$ on E, then chooses a random integer k. User B will send a pair of points $(k \cdot P, M + k \cdot Q)$, assuming $k \cdot Q \neq O$.

To decrypt the message, user A will use his secret key to compute: $M = (M + kQ) - d(kP)$.

A disadvantage of this cryptosystem is that one must use a point $M = (m_x, m_y)$ on the elliptic curve to embed the message m. Hence it limits the plaintext space somehow. This is another drawback, since the eavesdropper can recover the full message if he somehow knows only one part of it, either m_x or m_y.

An alternative version of this cryptosystem is to replace the elliptic curve addition in $(M + kQ)$ by a regular finite field addition in both encryption and decryption. This avoids the above disadvantage; now message M is not necessarily embedded on the elliptic curve.

Encryption: Assuming that $kQ \neq (0,0)$, then the ciphertext is the pair of data that includes: $M \oplus (kQ) = (m_x + (kQ)_x, m_y + (kQ)_y)$ and point (kP). Here the addition + is the regular finite field addition.

Decryption: Using his secret key a, user A can recover the plaintext as:

$[M \oplus (kQ)] - d(kP) = (m_x + (kQ)_x - (kQ)_x, m_y + (kQ)_y - (kQ)_y) = (m_x, m_y) = M$.

In this algorithm, one does not need to compute the order of the elliptic curve. But in practice, we need to do so for security confidence on the infeasibility of the ECDLP.

The ciphertext is expanded by a factor of 2 (or only 3/2 if one uses compressing techniques). The same drawback: if someone knows m_x (or m_y), he can solve for $(kQ)_x$ then $(kQ)_y$ and m_y (or m_x) easily.

Another version of this scheme is just to use the x-coordinate of a point, and hence the message M is now just written as $M = m_x$. Then the message will be XOR-ed with $(kQ)_x$ and concatenated with $k \cdot P$ to form the ciphertext: $kQ \parallel M \oplus (kQ)_x$. The decryption is similar to the method described above, after $(k \cdot P)$ is extracted from the ciphertext.

Alternative version using symmetric key encryption: The message will be encrypted by a symmetric key encryption scheme *SKE* with (k·Q) as a secret key and will be sent together with (k·P) to user A.

User A receives the ciphertext, the pair $e = SKE_{(kQ)}(M)$ and point (k·P). User A then uses his secret key a to compute a· (k·P) = k·Q and uses (k·Q) to decrypt the message: $M = SKE^{-1}_{(k.Q)}(e)$.

2.B.2. Massey-Omura elliptic curve cryptosystem ([MO86])

This cryptosystem will be used to send a message m that is embedded as a point P on a fixed elliptic curve whose order N is computed and also publicly known. Each user will choose at random an integer e such that gcd(e, N) = 1, and compute d such that $e.d \equiv 1 \pmod N$. Both e and d are kept secret. That is, for some integer k,

$$(ed)\cdot P = (1 + kN)\cdot P = P + kN\cdot P = P + O = P.$$

User A sends e_AP to user B. Then user B sends back $e_B(e_AP)$ to user A. Then user A does his decryption by computing: $d_A.(e_B(e_AP) = e_B.P$ and sends it back to user B. No one can read the message yet, up to this stage. Finally, user B does his decryption to read the message: $d_B(e_BP) = P$.

The security of this cryptosystem is obviously based on the infeasibility of ECDLP. Massey-Omura elliptic curve cryptosystem can be considered a variant version of the original Diffie-Hellman key exchange scheme.

2.B.3. Menezes-Vanstone elliptic curve cryptosystem ([MV90],[MV93])

This is also mentioned as a version of ElGamal cryptosystem in many cryptography literature. Let E be a non-supersingular elliptic curve defined over a finite field F_p, where prime $p > 3$. Choose a point P in $E(F_p)$ of order n and compute the point $Q = d$·P with the secret key d. The pair (P, Q) are public keys.

Encryption: The sender will choose a secret random number k in the interval [1, n –1]. The ciphertext of a message $m = (m_1, m_2) \in Z_p^* \times Z_p^*$ will be the triple including the point $k.P$ and two finite field elements y_1 and y_2 where $y_1 = c_1m_1 \pmod p$ and $y_2 = c_2m_2 \pmod p$, and assuming $kQ = (c_1, c_2) \neq (0,0)$.

Decryption: The receiver uses his secret key d to compute the point d·(k·P) that should be exactly $kQ = (c_1, c_2)$. Hence the receiver can recover the message by: $(y_1c_1^{-1} \pmod p, y_2c_2^{-1} \pmod p) = (m_1, m_2)$.

Analysis: There is also a message expansion of factor 2 (or 3/2 if one uses compressing techniques). The same drawback: if one knows m_x (or m_y), he can solve for m_y (or m_x) easily. To prevent this attack, one should send only k·P and one finite field element $y = c_1\, m \pmod p$.

There are other proposals/standards computing y_1 and y_2 in more complex algorithms from c_1, c_2, m_1 and m_2 in order to prevent an eavesdropper, who knows y_1, y_2 and half the message, say m_1, from recovering the other half message m_2 or from substituting m_1 by his own message.

2.C. Digital signature & authentication schemes

2.C.1. Elliptic curve digital signature schemes

a. Elliptic curve digital signature algorithm (ECDSA) – ANSI X9.62.

This is a version of ElGamal elliptic curve digital signature scheme and is an analogue to DSA, Digital Signature Algorithm. In this document, the hash function H is the Secure Hash Algorithm (SHA-1), whose output e is a 160-bit string.

Step 1. Initial Setup: All users will use the same underlying finite field F_q, where either $q = 2^m$ or q is a prime greater than 3. Let E be a non-supersingular elliptic curve defined over the finite field F_q. Choose P as a base point of prime order n in $E(F_q)$. We will work on the group $\langle P\rangle = \{O, P, 2P, \ldots, (n-1)P\}$.

Step 2. Key generation: The signer will select a statistically unique and unpredictable (or random) integer d in the interval $[1, n-1]$, and compute point $Q = d \cdot P$. The signer's private key is d and his public key is Q.

Step 3. Signature generation: The signer will select a statistically unique and unpredictable integer k in the interval $[1, n-1]$, and compute $kP = (x_1, y_1)$ and check that $r = x_1 \neq 0 \pmod n$. If $r = 0$, then select another k. Compute $e = H(M)$ and check that $s = k^{-1}(e + dr) \bmod n \neq 0$. If $s = 0$, then select another k. The signature for message M is a pair (r,s).

Step 4. Signature verification: The verifier will compute the values $e = H(M)$, $u_1 = es^{-1} \pmod n$ and $u_2 = rs^{-1} \pmod n$. Then compute the elliptic curve point $u_1P + u_2Q = (x_2, y_2)$, using the signer's public key Q, then the value $v = x_2 \pmod n$. The signature is accepted if $v = r$.

The security of this cryptosystem does not depend on the choice of the base point P as long as its order n satisfies the requirements: $n > 2^{160}$ (or n has at least 161 bits.)

Signature size: To achieve the same security level (in terms of MIPS years to break-in using the best attacks) of DSA (160-bit q and 1024-bit p), or RSA (1024-bit modulus n), the parameter n should have at least 161 bits. This could help ECDSA to resist against Vaudenay's attack.

Public key size: By the point compression technique, a point (x, y) on the elliptic curve can be represented simply by x-coordinate and a single bit of y-coordinate. Hence in the above case, a public key size is 161 bits only.

Note that the security of such schemes also depends on the security of hash function that is used. In current estimation, for short-term security, n should have at least 161 bits; for medium-term security, 180 bits. One always expect that these lower bounds should be increased as technology advances.

Brown [B00] proved that ECDSA is secure against existential forgery by adaptive chosen-message attack if the goup of points on the elliptic curve is modeled by a generic group and the hash function is collision-resistant. Nguyen & Shparlinski [NS01] discussed the insecurity of the ECDSA with partially known nonces.

b. Other schemes (Agnew, Mullin & Vanstone [AMV90])

The setup and key generation are similar to those in ECDSA described above. Instead of computing the inverse k^{-1} in the formula $s = k^{-1}(e + dr) \bmod n$, we define: $s = d^{-1}(e + kr) \bmod n$. Hence, we need to compute only one fixed inverse d^{-1} of the private key d for all messages. Recall that: $kP = (x_1, y_1)$, $r = x_1 \neq 0 \pmod n$, $e = H(M)$. The signature for message M is a pair (r, s).

To verify signature, we compute: $e = H(M)$, $u_1 = -er^{-1} \pmod n$ and $u_2 = sr^{-1} \pmod n$. Then using the signer's public key Q to compute: $u_1P + u_2Q = (x_2, y_2)$ and $v = x_2 \pmod n$. The signature is accepted if $v = r$.

The security of this scheme is based partially on the intractability of the following problem:

Given $Q = (x_Q, y_Q)$, *find a point* $P = (x_P, y_P)$, *such that* $s(x_P, y_P) = (x_Q, y_Q)$, *where* $s = (x_P \bmod n)$, *if such a point exists.*

This problem is thought to be more difficult than ECDLP. The existence is also not proven yet. This scheme is basically similar to a general version of Nyberg-Rueppel's signature scheme with message recovery.

Modified ElGamal digital signature schemes are discussed in Saryazdi [S90], Horster, Michels & Petersen ([HMP94],[HMP94a]), and He & Kiesier [HK94].

c. Schnorr signature scheme

Schnorr [S91] proposed another modification of the ElGamal scheme to avoid the inverse computation. It also requires a hash function $H(M, x)$, for a message M using a key x. Its signature can be smaller than the ElGamal scheme. The setup and key generation steps are similar as those steps in ECDSA described above.

Signature generation: Select a random integer k in the interval $[1, n-1]$, then compute the point $kP = (x_1, y_1)$. Then compute $e = H(M, x_1)$ and $s = (k + de) \bmod n$. The signature for message M is a pair (e, s).

Signature verification: Compute the point $sP - eQ = (z, t)$, using the signer's public key Q. The signature is accepted if $H(M,z) = e$.

The Schnorr scheme is secure against passive attacks but not yet known for active attacks. In order to resist against some known attacks, the hash function $H(M, x)$ used in this scheme must satisfy two basic conditions: $H(M, x)$ must be almost uniformly distributed with respect to variable x, and be a one-way function with respect to variable M. Otherwise, for fixed message M and value e, if the hash function $H(M, x)$ is not uniform with respect to x, one can compute the point $(a, b) = sP - eQ$, for a random s until the equality $e = H(M, a)$ holds. This attack yields a signature (s, e) for the given message M.

There is another case: chosen message attack. One can choose an arbitrary signature (s, e) and compute the point $(z, t) = sP - eQ$. Then he can solve for message M from the equation $e = H(M, z)$, if the hash function H is not a one-way function with respect to M. The hash function is not required to be collision-free with respect to M. If we have: $H(M, x) = H(M', x)$, the signature for message M cannot be used to sign message M' since it depends also on a random number k, hence on a random point $kP = (x, y)$.

2.C.2. Elliptic curve digital signature schemes with message recovery

a. Nyberg-Rueppel's signature scheme with message recovery ([NR96])

Signature generation: The signer will select a random integer k in the interval $[1, n-1]$, then compute the point $R = kP = (x_1, y_1)$. This is called a one-time key pair (k, R). Then the signer will compute e and check that $e = (x_1 + M) \bmod n \neq 0$. If $e = 0$, the signer must select another k and repeat. Compute the integer $s = k - de \pmod n$. Then the signature for message M is a pair (e,s).

Signature verification: One cannot verify directly the signature in this scheme. The signature is accepted if the message is recovered properly. This can be achieved by adding redundancy to the message before it is signed and checking the redundancy after it is recovered.

Message recovery: Compute the point $sP + eQ = (z, t)$. The message is decrypted by $M = (e - z) \bmod n$.

Nyberg-Rueppel's scheme is similar to Schnorr's scheme in which the hash function H is replaced simply by: $H(M + x_1) = M + x_1$. In Nyberg-Rueppel's scheme, we are interested in recovering the message embedded in the signatures without verifying the signature itself. In Schnorr's scheme, the receiver can attain the message somehow and use it to verify the signature.

An alternative scheme

Signature generation: Embed the message into a point M on the elliptic curve. Select a random integer k in the interval $[1, n - 1]$; then compute the point $R = kP + M = (x_1, y_1)$. The signature is a pair (R,s) where $s = k - dx_1 \pmod n$.

Signature verification: One also cannot verify the signature in this scheme. The signature is accepted if the message is recovered properly.

Message recovery: From $R = (x_1, y_1)$, one can compute the point $(sP + x_1Q)$. Then the message is recovered by computing the point: $(sP + x_1Q) - R = M$.

In a more general term, in Nyberg-Rueppel's schemes, the integer s satisfies a signature equation $Ad + Bk + C = 0 \pmod n$, where (A, B, C) is a fixed permutation of $(\pm x_1, \pm s, \pm 1)$. Nyberg & Rueppel [NR96] also discussed details on alternative schemes.

Nyberg-Rueppel's scheme can be also used as a key agreement algorithm in the multiplicative group setup. El Mahassni, Nguyen & Shparlinski [MNS01] discussed the insecuriry of Nyberg-Rueppel schemes with partially known nonces.

b. Vanstone's signature scheme with message recovery

It is another modified version of the ElGamal scheme and is also an enhancement of Nyberg-Rueppel's scheme. The setup and key generation steps are similar to those steps in ECDSA described above. In place of the hash function H, one uses a symmetric key encryption/decryption scheme, E_x and D_x in order to recover the encrypted message.

Signature generation: Select a random integer k in the interval $[1, n-1]$, where the prime n is order of the base point P, as usual. Then compute the point $kP = (x', y')$. Then compute: $e = E_{x'}(M)$, $e' = H(e)$ and $s = (k + de') \bmod n$. The signature for message M is a pair (e, s).

Signature verification: One cannot verify directly the signature in this scheme. The signature is accepted if the message is recovered properly. This can be achieved by adding redundancy to the message before it is signed and checking the redundancy after it is recovered.

Message recovery: Compute an integer $e' = H(e)$ and the point $sP - e'Q = (z,t)$. The message is recovered as: $D_z(e) = M$.

In fact, because of the relation $(z,t) = (x', y')$, as the same point on the elliptic curve, the encryption E and decryption D, that used x-coordinate as a secret key, can be designed generally as a symmetric key encryption scheme with the key (kP):

$$E = SKE_{(kP)}(M) = e \text{ and } D = SKE^{-1}_{(kP)}(e) = M.$$

Signature size: This scheme produces a signature of the size equal to the sum of message size and elliptic curve size (or order n).

2.C.3. Summary of digital signature schemes

Digital Signature Schemes	Signature Generation	Signature Verification	Message Recovery

ECDSA (ANSI X9.62)	$R = kP = (x_1, y_1)$ $r = x_1 \neq 0 \pmod n$ $e = H(M)$ $s = k^{-1}(e + dr) \bmod n$ Signature: (r, s)	$e = H(M)$ $u_1 = es^{-1} \pmod n$ $u_2 = rs^{-1} \pmod n$ $u_1P + u_2Q = (x_1, y_1)$ $v = x_1 \pmod n$ Accept the signature if $v = r$	No. Message is encrypted and then decrypted in a separate scheme.
Agnew, Mullin & Vanstone's scheme ([AMV90])	$R = kP = (x_1, y_1)$ $r = x_1 \neq 0 \pmod n$ $e = H(M)$ $s = d^{-1}(e + kr) \bmod n$ Signature: (r, s)	$e = H(M)$ $u_1 = -er^{-1} \pmod n$ $u_2 = sr^{-1} \pmod n$ $u_1P + u_2Q = (x_1, y_1)$ $v = x_1 \pmod n$ Accept the signature if $v = r$	No. Message is encrypted and then decrypted in a separate scheme.
Schnorr's scheme	$R = kP = (x_1, y_1)$ $e = H(M, x_1)$ $s = (k + de) \pmod n$ Signature: (e, s)	$sP - eQ = (z,t)$ Accept the signature if $H(M,z) = e$.	No. Message is encrypted and then decrypted in a separate scheme.
Vanstone's scheme	$e = E_{x'}(M)$ $e' = H(e)$ $s = (k + de') \pmod n$ Signature: (e, s)	Not directly. The signature is accepted if the message is recovered properly.	$e' = H(e)$ $sP - e'Q = (z,t)$ Accept the message if $D_z(e) = M$
Nyberg-Rueppel's scheme 1	$e = (x_1 + M) \bmod n$ $s = (k - de) \pmod n$ Signature: (e, s)	Not directly. The signature is accepted if the message is recovered properly.	$sP + eQ = (z,t)$ Message: $M = (e - z) \bmod n$
Nyberg-Rueppel's scheme 2 (alternative)	$R = kP + M = (x_1, y_1)$ $s = k - dx_1 \pmod n$ Signature: (R, s)	Not directly. The signature is accepted if the message is recovered properly.	Message: $(sP + x_1Q) - R = M$

Table 2.1. Summary of digital signature schemes

Initial setup & Key generation

All users will use the same underlying finite field F_q, where either $q = 2^m$ or q is a prime greater than 3. Let $E(F_q)$ be a non-supersingular elliptic curve defined over a finite field F_q. Choose P as a base point of prime order n in $E(F_q)$. Each user selects a random integer d in the interval $[1, n - 1]$, then computes the point $Q = dP$. His private key is d and public key is Q.

For all schemes, the sender always selects a random integer k in the computation. Both sides need a good hash function H.

2.C.4. Signcryption schemes

The original "signcryption" schemes were propsed Zheng and Imai's works, [Z98] and [IZ98]. It is a cryptographic method that fulfils both functions of encryption and digital signature with a smaller cost than the cost required by a typical signature-then-encryption procedure.

Signcryption: The signer will select a random integer k in the interval $[1, n-1]$, and compute $kP = (x_1, y_1)$, $r = H(M, kP)$ and $s = k(r + d)^{-1}$. The signature is (r, s).
Unsigncryption: Compute the elliptic curve point $K = s(rP + Q)$, using the signer's public key Q, then check whether $H(M, K) = r$.

In the other version, compute $s = k(1 + rd)^{-1}$ in signcryption step and compute $K = s(P + rQ)$ in unsigncryption step.

2.C5. Schnorr's authentication schemes ([S91])

The setup and key generation steps are similar to those steps in the ECDSA described above or in Schnorr's signature scheme.

Initiation: Prover A selects a random integer k in the interval $[1, n-1]$ then computes the point $kP = (x_1, y_1)$. Then prover A uses the hash function H to compute the value $h = H(0, x_1)$. Prover A sends to verifier B its identification string I, its public key Q, the signature S for the pair (I, Q) and h. Verifier B then verifies the signature S and send a random integer $e \in [1, n-1]$ to A. Prover sends to B: $s = k + de \pmod n$.

Verification: Verifier B verifies the pair (I, Q) either by checking the signature S or computing $sP - eQ = (z, t)$ and checking that $H(0, z) = h$.

An attack C can cheat by guessing e and sending to verifier B a wrong proof: $h = H(0, x_1)$ where $(x_1, y_1) = kP - eQ$, for a random integer k and $s = k$. The probability for successful guessing e is only $(n-1)^{-1}$. Verifier B can also choose e freely in order to learn prover A's method of authentication.

2.D. Key agreement & key exchange schemes

In a key agreement or key exchange scheme, every party contributes information/data in order to compute a session key. While in a key transfer scheme, one party sends the computed session key to the other parties in a secure way.

2.D.1. Elliptic curve Diffie-Hellman key agreement ([DH76])

Two users A and B first agree on a base point P on an elliptic curve E. User A will choose a random number a, compute $a \cdot P$, then send $a \cdot P$ publicly to user B, while keeping a secret. User B will choose a random number b, compute $b \cdot P$ then send $b \cdot P$ publicly to user A, while keeping b secret.

Both users now can compute the common secret key $(ab \cdot P)$ by one user's secret key and other's public key.

For user A, it is the point $a \cdot (b \cdot P)$; for user B, it is the point $b \cdot (a \cdot P) = a \cdot (b \cdot P)$.

Anyone else could know the public information P, aP, and bP, but it is infeasible to find the secret keys a, b and $(ab \cdot P)$ (by the property of ECDHP.)

Both ElGamal and Massey-Omura cryptosystems are variants of Diffie-Hellman key exchange scheme.

2.D.2. Elliptic curve MTI key agreement (Imai, Matsumoto & Takashima, [IMT86])

Each user has a key pair. User A has (a, Q_A), where $Q_A = aP$ and user B has (b, Q_B), where $Q_B = bP$. User A selects a random integer k_A, then computes and sends to user B the point $K_A = k_A . Q_B$. Similarly, user B will send to A the point $K_B = k_B . Q_A$, where k_B is a random integer. Each user now can compute the shared secret value K by:

For user A, it is $K = a^{-1} k_A K_B = a^{-1} k_A k_B \, aP = k_A k_B P$,

For user B, it is $K = b^{-1} k_B K_A = b^{-1} k_B k_A \, bP = k_A k_B P$.

An alternative version: User A selects a random integer k_A then computes and sends to user B two points $K_A = k_AP$ and Q_A. Similarly, user B will send to A two points $K_B = k_BP$ and Q_B. Each user now can compute the shared secret value K by:

For user A, it is $K = aK_B + k_AQ_B = (ak_B + k_Ab)P$.

For user B, it is $K = bK_A + k_BQ_A = (bk_A + k_Ba)P$.

In general, these schemes are vulnerable to a few active attacks, including unknown key-share attacks.

2.D.3. Elliptic curve Menezes-Qu-Vanstone key agreement ([MQV95])

Setup: One will construct an appropriate elliptic curve E as usual, such that n is a prime factor of its order $\#E$ (usually, it is order of the base point P on E.) Let us define: $h = \lceil (\log_2 n)/2 \rceil$.

User A has two key pairs (a_1, A_1) and (a_2, A_2), where private keys $a_1, a_2 \in [1, n - 1]$ and public keys $A_1 = a_1P$ (that is called long term or static key) and $A_2 = a_2P = (x, y)$ (which is called the short term or ephemeral key.) User B also has two key pairs (b_1, B_1) and (b_2, B_2), where $B_1 = b_1P$ and $B_2 = b_2P = (x', y')$, with two secret keys $b_1, b_2 \in [1, n - 1]$. Let $t_A = f(A_2) = f(x, y) = (x \bmod 2^h) + 2^h$ and $t_B = f(B_2) = f(x', y') = (x' \bmod 2^h) + 2^h$. Then two users will do computations using one's own secret keys and the other's public keys.

User A will compute	User B will compute
an integer $e_A = (t_Aa_1 + a_2) \bmod n$	an integer $e_B = (t_Bb_1 + b_2) \bmod n$
a point $R_A = e_A(t_BB_1 + B_2)$.	a point $R_B = e_B(t_AA_1 + A_2)$.
Check if $R_A = (x_A, y_A) \neq O$.	Check if $R_B = (x_B, y_B) \neq O$.

Table 2.2. Computations in the MQV key exchange scheme

Then $K = x_A = x_B$ is the shared secret value or shared key for both users A and B.

Verification: One can observe simply that $a_iB_j = a_ib_jP = b_jA_i$, for $i, j = 1, 2$. Hence it is easy to verify the scheme:

$$R_A = e_A(t_BB_1 + B_2) = [(t_Aa_1 + a_2) \bmod n]\,(t_BB_1 + B_2)$$
$$= [(t_Aa_1 + a_2) \bmod n]\,[(t_Bb_1 + b_2) \bmod n]P, \text{ (since point } P \text{ has order } n)$$
$$= [(t_Bb_1 + b_2) \bmod n]\,(t_AA_1 + A_2) = e_B(t_AA_1 + A_2) = R_B.$$

Note that in this scheme, one needs both x- and y-coordinates of elliptic curve points in computation. The special feature of this scheme requires both public and private keys of both parties implied in the signatures and their verifications. Hence the MQV scheme can prevent efficiently the man-in-the-middle attack. Refer also to Law, Menezes, Qu, Solinas & Vanstone [LMQSV98].

2.E. RSA-type elliptic curve cryptosystems

We now mention RSA-type elliptic curve cryptosystems may be both controversial and interesting to many researchers. There are works on designing such cryptosystems or exploiting the connections with RSA cryptosystems.

2.E.1. Public key cryptosystem using elliptic curves over a ring Z_N

a. Elliptic curves over a ring Z_N

The elliptic curve over Z_N is of the form: $E_{a,b}$: $y^2 = x^3 + ax + b$, where $a, b \in Z_N$, and $\gcd(4a^3 + 27b^2, N) = 1$. In general, over a ring Z_N, the set of points in $E_{a,b}(Z_N)$ (including the point at infinity denoted by O_N) does not form a group. The same addition rules defined for an elliptic curve over a finite field cannot be extended to the ring Z_N.

The simple reason is that the inversion of a non-zero number n works in modulo a prime p but does not work in modulo a composite number N, if $\gcd(n, N) > 1$.

Under modulo p, for p being some prime factor of N, the elliptic curve $E_{a,b}$ over Z_N is reduced to an elliptic curve $E_{\bar{a},\bar{b}}$ over the prime finite field F_p, where we denote $\bar{a} = \bar{a}_p = a \pmod p$. Let $P = (x, y) \in E(Z_N)$, then $P_p = (\bar{x}, \bar{y}) \in E(F_p)$, and $(O_N)_p = O_p$, the point at infinity of $E(F_p)$. We may drop subscripts a and b where there is no confusion.

From now on, we are interested in the case that N is a product of two distinct odd big primes p and q. For convenience, we also use the notation q for a prime factor of the composite number N. Do not confuse with the subscript q for finite fields where $q = p^m$. By the Chinese remainder theorem, any $x \in Z_N$ can be represented uniquely as a pair $(\bar{x}_p, \bar{x}_q) \in F_p \times F_q$. We consider the product $\tilde{E}(Z_N)$ of two groups $E(F_p)$ and $E(F_q)$. Hence $\tilde{E}(Z_N)$ is a group. We have $\tilde{E}(Z_N) = E(F_p) \times E(F_q) = \{(P_p, P_q)$, where $P_p = (\bar{x}_p, \bar{y}_p) \in E(F_p)$ and $P_q = (\bar{x}_q, \bar{y}_q) \in E(F_q)\}$ and $O_N = (O_p, O_q)$.

For the elliptic curve $E(Z_N)$, we observe that each point $P \in E(Z_N)$ corresponds to a unique element $(P_p, P_q) \in \tilde{E}(Z_N)$, except for elements in which exactly one of the points (P_p or P_q) in the pair is the point at infinity. Then we have $\# \tilde{E}(Z_N) = \#E(F_p) \times \#E(F_q)$. The number of elements in the set $E(Z_N)$ - which is a subset of $\widetilde{E}(Z_N)$ - is easily computed by: $\#E(Z_N) = (\#E(Z_p) - 1).(\#E(Z_q) - 1) + 1 = \#\widetilde{E}(Z_N) - \#E(Z_p) - \#E(Z_q) + 2$. This number was not used anywhere in the cryptosystems. Instead we are interested in

$$\#\tilde{E}(Z_N) = \#E(F_p).\#E(F_q) \text{ or } M_N = \operatorname{lcm}(\#E(F_p), \#E(F_q)).$$

b. Addition rule and factorization algorithm

Also let $Q = (x', y') \in E(Z_N)$ corresponding to a unique element $(Q_p, Q_q) \in \tilde{E}(Z_N)$. The addition operation on $E(Z_N)$ is defined by the component-wise addition in each group of the product group $\tilde{E}(Z_N)$. That is, $P + Q = (P_p + Q_p, P_q + Q_q)$. Particularly, we have the scalar point multiplication formula: $kP = (kP_p, kP_q)$, for any integer k.

The points (P_p, O_q) and (O_p, P_q), for any P_p or P_q that is not the point at infinity, are called non-realizable points. They cannot be the result of adding any two points on $E(Z_N)$ If p and q are very large primes, then the percentage of non-realizable points is negligible.

In fact, the addition defined for $E(Z_N)$ above is undefined if and only if the resultant is a non-realizable point of the form $[P_p, O_q]$ or $[O_p, P_q]$. When the point addition would result in non-realizable points with non-negligible probability, one could have a feasible integer factoring algorithm for N. In other words, one can claim:

Lemma: *If P and Q are two points on $E(Z_N)$ whose addition is undefined, then the knowledge of points P and Q is sufficient to factor N.*

In the next section, we will discuss public key cryptosystems, digital signature and key agreement schemes based on elliptic curves over a ring Z_N. We will use the following result that can be proved simply.

Lemma: *If $M_N = \operatorname{lcm}(\#E(F_p), \#E(F_q))$, then for any point $P \in E(Z_N)$, and any integer k, we have the identity: $(kM_N + 1)P = P$.*

As the RSA cryptosystem, two distinct large primes p and q are kept secret and the modulus $N = pq$ is publicly known. The security of RSA-type elliptic curve

cryptosystems is based on the difficulty of factoring modulus N. The group orders $\#E(F_p)$ and $\#E(F_q)$ are served as the trapdoor.

The public key e is chosen to be relatively prime to both $\#E(F_p)$ and $\#E(F_q)$. Then e is also relatively prime to $M_N = \text{lcm}(\#E(F_p), \#E(F_q))$. That is: gcd $(e, M_N) = 1$. Then the secret key can be defined as: $d = e^{-1} (\text{mod } M_N)$. Hence we have $(ed) \cdot P = P$.

2.E.2. RSA-type elliptic curve cryptosystems

a. KMOV elliptic curve cryptosystem

Three KMOV cryptosystems were proposed by Koyama, Maurer, Okamoto & Vanstone [KMOV92]. As an analogue of the RSA cryptosystem, the security of these systems are based on the difficulty of factoring n.

Type 0 scheme requires computation of the order of a general elliptic curve. It can be used only in digital signature schemes. The signature is about twice the message. To sign a message M, one embeds it as a point $P = (x, y) \in E(Z_N)$, and the signature is the point $Q = (u, v) = d \cdot P$, using his private key. To verify the signature, the receiver computes the point $P = (x, y) = e \cdot Q$, using the sender's public key, and extracts the message M. The drawback of this system is also common for all cryptosystems using elliptic curves with large primes p, the Schoof's algorithm to compute the order $\#E(F_p)$ is still infeasible. This scheme cannot be used for a public-key cryptosystem, since knowledge of the trapdoor is required to create a point on $E(Z_N)$, that corresponds to a message.

Type 2 scheme is a Rabin-type generalization of type 1, where the public key $e = 2$. It also has the 4-ambiguity in decrypted message, as in the original Rabin scheme.

Type 1 scheme is the most practical of three schemes. To set up, user A has the modulus $N = pq$ where p and q are two distinct large primes, which are kept secret and satisfy either case of these two special cases:

Case $p \equiv q \equiv 2 \pmod 3$: The elliptic curve $E(Z_N)$ is of the form $E_{0,b}$: $y^2 = x^3 + b$, where coefficient b is determined by the message, $m = (m_x, m_y) \in Z_n \times Z_n$, to be encrypted: $b \equiv m_y^2 - m_x^3 \pmod N$. (Or also by the ciphertext: $b \equiv c_y^2 - c_x^3 \pmod N$ as we will observe below.) But we do not need b explicitly in the computation.

Case $p \equiv q \equiv 3 \pmod 4$: The elliptic curve E is of the form $E_{a,0}$: $y^2 = x^3 + ax$, where a must be computed by the sender using the message or plaintext $m = (m_x, m_y)$, $a \equiv (m_y^2 - m_x^3) m_x^{-1} \pmod N$, or by the receiver by the encrypted message or ciphertext $c = (c_x, c_y)$, $a \equiv (c_y^2 - c_x^3) c_x^{-1} \pmod N$.

Key generation: In both cases, the elliptic curve orders are easily computed, $\#E(F_p) = p + 1$ and $\#E(F_q) = q + 1$. Let $M_N = \text{lcm}(p + 1, q + 1)$ that is also kept secret. His public key will be N and e, where e is randomly selected such that $\gcd(e, M_N) = 1$. His private key will be d, such that $ed \equiv 1 \pmod{M_N}$.

Encryption: To send a message to user A, user B will encrypt a message $m = (m_x, m_y)$ using user A's public key e: $e.(m_x, m_y) = (c_x, c_y)$ over $E(Z_N)$

Decryption: User A uses his secret key d to recover the message: $d.(c_x, c_y) = (m_x, m_y)$ over $E(Z_N)$.

b. Demytko's elliptic curve cryptosystem ([D94])

Like KMOV schemes, this cryptosystem is defined over a ring Z_N. It uses only the x-coordinate of a point on an elliptic curve. Its security is also based on the difficulty of

factoring N, where $N = pq$ and p and q are two secret large primes. The elliptic curve E is of the form E: $y^2 = x^3 + ax + b$ and gcd $(4a^3 + 27b^2, N) = 1$. Let E_p denote the elliptic group modulo p of elements (x, y) satisfying the equation $y^2 = x^3 + ax + b \pmod p$ together with the point at infinity O. Its order is $\#E_p = p + 1 - t_p$. Then its twist curve E'_p has order $\#E'_p = p + 1 + t_p$. We do the same for the other prime number q.

Key generation: His public key will be a pair N and e. There are four secret keys d_1, d_2, d_3 and d_4 for decryption, which are defined by $ed_i \equiv 1 \pmod{N_i}$, for $1 \le i \le 4$ and the moduli N_i are defined as in the table 2.3.

Encryption: The message M is embedded as an x-coordinate of a point P on the elliptic curve E. We write $eP = ((eP)_x, (eP)_y)$. The ciphertext C will be the x-coordinate of the point (eP): $C = (eP)_x$.

Decryption: First, we compute: $u = C^3 + aC + b \pmod N$. Then we determine whether u is a quadratic residue modulo p and/or modulo q and refer to the table 2.3. for the right decryption key (secret key) d_i.

We have the decrypted message as: $M = (d_i.(C,*))_x \pmod N$, and we can do so without using the y-coordinate.

$u = C^3 + aC + b \pmod N$		Modulus N_i	Decryption
$\left(\frac{u}{p}\right) = 1$	$\left(\frac{u}{q}\right) = 1$	$N_1 = \text{lcm}\,(p + 1 - t_p, q + 1 - t_q)$	$M = (d_1.(C, *))_x \pmod N$
$\left(\frac{u}{p}\right) = 1$	$\left(\frac{u}{q}\right) = -1$	$N_2 = \text{lcm}\,(p + 1 - t_p, q + 1 + t_q)$	$M = (d_2.(C, *))_x \pmod N$
$\left(\frac{u}{p}\right) = -1$	$\left(\frac{u}{q}\right) = 1$	$N_3 = \text{lcm}\,(p + 1 + t_p, q + 1 - t_q)$	$M = (d_3.(C, *))_x \pmod N$
$\left(\frac{u}{p}\right) = -1$	$\left(\frac{u}{q}\right) = -1$	$N_4 = \text{lcm}\,(p + 1 + t_p, q + 1 + t_q)$	$M = (d_4.(C, *))_x \pmod N$

Table 2.3. Demytko's elliptic curve cryptosystem: moduli and decryptions

Knowing prime numbers p and q, it is easy to observe whether the modular equations $x^2 \equiv u \pmod p$ and $x^2 \equiv u \pmod q$ have solutions and find those solutions. But finding a solution for $x^2 \equiv u \pmod N$ is a much more difficult problem. It is claimed to be equivalent to factoring N. One must know p and q to compute N_i's. The decryption time can be reduced by a factor of 4 if one computes M modulo p and q then combine the result using the Chinese remainder theorem. This scheme is also used for digital signatures where a sender uses one of his secret keys d_i to sign a message and the receiver uses the sender's public key to verify the signature.

c. Koyama & Kuwakado's elliptic curve cryptosystem

Koyama & Kuwakado [KK94] proposed an elliptic curve cryptosystem using the elliptic curves of the form: $E_N(a, 0)$: $y^2 = x^3 + ax$, $a \not\equiv 0 \pmod p$ or $E_N(0, b)$: $y^2 = x^3 + b$, $b \not\equiv 0 \pmod p$ over a ring Z_N, where $N = pq$. This can be considered a special case of Demytko's scheme. It is also a complement to KMOV case in term of restriction on prime numbers p and q.

Case 1: $E_N(a, 0)$: $y^2 = x^3 + ax$, $a \not\equiv 0 \pmod p$

If $p \equiv q \equiv 3 \pmod 4$, it is a case of KMOV cryptosystem.

If $p \equiv q \equiv 1 \pmod 4$, we have the formulae to find the order of the elliptic curve that is non-supersingular. The algorithm discussed in chapter 4 will show all four possible values of these orders. Then the Koyama & Kuwakado scheme is similar to KMOV for the case $p \equiv q \equiv 3 \pmod 4$.

Key generation: The public key e can be chosen universally such that gcd (e, $\#E_p$) = gcd (e, $\#E_q$) = 1, for all 4 possible values of each order $\#E_p$ and $\#E_q$.The private keys are computed to satisfy conditions $ed_p \equiv 1 \pmod{\#E_p}$ and $ed_q \equiv 1 \pmod{\#E_q}$ for each particular elliptic curve that will be used.

Private keys depend on the message (plaintext and ciphertext). In order to have a message-independent private key, we can modify a little (to make it similar to KMOV scheme): $ed_p \equiv 1 \pmod{L_p}$, where L_p is the least common multiplier of all 4 possible values of orders $\#E_p$.

The plaintext $m = (m_x, m_y)$ that is embedded as an elliptic curve point such that the coefficient $a \equiv (m_y^2 - m_x^3).m_x^{-1} \pmod{N}$ must satisfy the condition $a \not\equiv 0 \pmod{N}$.

Encryption will be performed on the elliptic curve E_N (a, 0): $y^2 = x^3 + ax$. The ciphertext is: $C = (c_x, c_y) = e.(m_x, m_y)$.

Decryption: From ciphertext C, the receiver computes $a_p \equiv (c_y^2 - c_x^3)c_x^{-1} \pmod{p}$ and $a_q \equiv (c_y^2 - c_x^3)c_x^{-1} \pmod{q}$ to determine the proper elliptic curve and the associated private keys to use for decryption: $m_p = d_p.(c_x, c_y)$ over $E_p(a_p, 0)$ and $m_q = d_q.(c_x, c_y)$ over $E_q(a_q, 0)$. Then using the Chinese remainder theorem, the receiver can obtain the message $m = (m_x, m_y)$ from m_p and m_q.

If $p \equiv -q = 1 \pmod 4$, then the elliptic curve modulo q is supersingular and its order is $(q + 1)$. The scheme is the same as in the previous case, except we fix the value $\#E_q = q + 1$.

Case 2: $E_N(0, b)$: $y^2 = x^3 + b$, $b \not\equiv 0 \pmod{p}$

If $p \equiv q \equiv 2 \pmod 3$, it is a case of KMOV cryptosystem.

If $p \equiv q \equiv 1 \pmod 3$, there are formulae to find the order of the elliptic curve that is non-supersingular. The algorithm discussed in chapter 4 will show all 6 possible values of these orders. Then Koyama & Kuwakado scheme is similar to KMOV for the case $p \equiv q \equiv 2 \pmod 3$, where b will be computed from the plaintext $b \equiv m_y^2 - m_x^3 \pmod{N}$ and from the ciphertext $b_p \equiv c_y^2 - c_x^3 \pmod{p}$ and $b_q \equiv c_y^2 - c_x^3 \pmod{q}$.

If $p \equiv -q = 1 \pmod 3$, again, the scheme is the same as in the previous case, except we fix the value $\#E_q = q + 1$.

d. Elliptic curve cryptosystem of Meyer-Müller ([MM96])

Let N be a publicly known product of two large secret primes p and q, and $p \equiv q \equiv 11 \pmod{12}$. We use the elliptic curve of the form E: $y^2 = x^3 + ax + b$ over a ring Z_N and satisfying the condition gcd $(4a^3 + 27b^2, N) = 1$.

Encryption: The sender chooses randomly $0 \neq r \in Z_N$, and embeds the message m into a point $P = (m^2, rm^3)$ on E where $a = r^3$ and $b = (r^2 - 1)m^6 - am^2$. Check that gcd$(4a^3 + 27b^2, N) = 1$. Otherwise, if gcd$(4a^3 + 27b^2, N) > 1$, then a factor of N can be found.

Then the sender computes $Q = 2.P = (x_Q, y_Q)$. Let l = lsb (y_Q) and type t be represented by a Jacobi symbol $t = \left(\frac{y_Q}{N}\right)$. The ciphertext will consist of (a, b, x_Q, t, l).

Decryption: The receiver computes the unique square root y_Q of $(x_Q^3 + ax_Q + b)$, with type t and the least significant bit l. Let $Q = (x_Q, y_Q)$. Then the receiver computes

the index set $I = \{1 \le i \le s \mid 2P_i = Q, a^2 = y_{P_i}^6 . x_{P_i}^{-9}\}$. The set I must have only one point P_I, and the message is $m = y_{P_I}^3 x_{P_I}^{-4} a^{-1}$.

The security of this cryptosystem is based on the intractability of solving "square root" (or half-point problem) in $E(Z_N)$. Joye & Quisquater ([JQ95],[JQ96]) showed that one can determine two polynomials $f(x)$, $g(x) \in Z_N[x]$ such that m^2 is a root, by using the expressions of a, b and coordinates of two points P and Q on E. This can be done simply, since these coordinates are functions of m. Then m^2 will be a root of the polynomial $h(x)$ = gcd ($f(x)$, $g(x)$). There is a very high probability that we can choose $f(x)$ and $g(x)$ such that $h(x)$ is of degree 1. Hence we can solve trivially for the root m^2. This cryptosystem is equivalent to the Rabin-Williams cryptosystem because it enables the user to recover the value of square of a message. Hence, it is equivalent to a factorization problem. Since m^2 has 4 roots modulo N, we can decrypt the message m simply from y_Q and its type t and the least significant bit l.

Signature scheme: To sign a message m, we hash a message m by $m' = H(m)$. Then we find integers r and $d \in Z_N^*$ such that for points $P = (d^2, rd^3)$ and $Q = 2 \cdot P$ on an elliptic curve of the form E: $y^2 = x^3 + r^3 x + b$. We will get $x_Q = m'$ and y_Q has type $t = 1$. The signature for m is (d, r).

Verification: Let $P = (d^2, rd^3)$ and compute $a = r^3$ and $b = (r^2 - 1)d^6 - ad^2$. Compute $Q = 2.P$ on the elliptic curve E: $y^2 = x^3 + ax + b$. The signature is accepted if y_Q has type $t = 1$, and $x_Q = m'$ and $m' = H(m)$.

e. Elliptic curve cryptosystem of Chua-Ling

For further interest, Chua & Ling [CL97] proposed a special cryptosystem using singular cubic curves instead of standard elliptic curves. Chua-Ling cryptosystem used the similar idea of Meyer-Müller on the singular cubic curve of the form C: $y^2 = x^3 + bx^2$ over a ring Z_N, where N is a publicly known product of two large secret primes p and q, and $p \equiv q \equiv 11 \pmod{12}$. (Notice that the cubic polynomial has multiple roots; hence the discriminant Δ vanishes.)

Encryption: The sender chooses randomly $r \in Z_N \setminus \{0, \pm 1\}$ and embeds the message m into a point $P = (m^2, rm^3)$ on C. Then the sender computes $Q = 2P = (x_Q, y_Q)$, $a = r^3$, $b = (r^2 - 1)m^2$, $l = \text{lsb}(y_Q)$ and $t = \left(\frac{y_Q}{N}\right)$. The ciphertext will consist of (a, b, x_Q, t, l).

Decryption: The receiver computes the unique square root y_Q of $(x_Q^3 + bx_Q^2)$ with type t and lsb l. Letting $Q_p = Q \pmod p$, the receiver then computes $I_p = \{1 \le i \le 2 \mid 2P_{p,i} = Q_p$ and $a^2 = y_{Pp,i}^6 x_{Pp,i}^{-9}\}$ and $m_p = y_{P_{p,i}}^3 x_{P_{p,i}}^{-4} a^{-1} \pmod p$. Similarly, the receiver gets m_q from the equation: $Q_q = Q \pmod q$. The receiver can have m by using the Chinese remainder theorem such that $m = m_p \pmod p$ and $m = m_q \pmod q$

Joye & Quisquater [JQ95a] showed that one needs the expressions of x_Q and a, b to find out 2 polynomials $f(x)$ and $g(x)$ of degree 2 and 3. Hence, with very high probability, one can solve for m^2 and then message m.

f. Paillier's elliptic curve encryption scheme

Paillier ([P99], [P00]) proposed three encryption schemes, which are defined on an elliptic curve over a ring Z_N, where $N = pq$, is the product of two large odd prime numbers. Over each finite field F_p and F_q, the elliptic curve E: $y^2z = x^3 + axz^2 + bz^3$ is the

quadratic twist of an anomalous elliptic curve. Hence, the elliptic curve orders are: $\#E(F_p) = p + 2$ and $\#E(F_q) = q + 2$. Therefore, $\#E(Z_N) = (p + 2)(q + 2)$.

For one encryption scheme, Paillier used the elliptic curve over the ring $R = Z/p^2qZ$. Then he computed: $\#E(Z/p^2qZ) = (p + 2)p \times \#E(F_q)$. The twist of this elliptic curve has the order: $\#E^{(d)}(Z/p^2qZ) = p^2 \times \#E^{(d)}(Z/qZ)$, where $\left(\frac{d}{p}\right) = -1$.

For another encryption scheme, Paillier used the elliptic curve over the ring $R = Z/N^2Z$. Then he computed: $\#E(Z/N^2Z) = (p + 2)(q + 2)pq$.

Galbraith [G02] proposes a generalization of Paillier schemes over elliptic curves on rings Z_N, where $N = pq$, a product of two large odd primes.

$$E: y^2z = x^3 + axz^2 + bz^3, \text{ where } \gcd(N, 6(4a^3 + 27b^2)) = 1.$$

Let $M = \text{lcm}(\#E(F_p), \#E(F_q))$. The user needs a point $Q = (x : y : z)$ that has order dividing M in the elliptic curve $E(Z/N^2Z)$. That point can be of the form $Q = NQ'$, where Q' is any ramdom number. The public key will be N, a, b and Q. The secret key will be M.

Encryption: A message m is embedded in Z_N as usual. Let $P_m = (mN : 1 : 0)$. The sender chooses a random number r and sends the point $S = rQ + P_m$ to the receiver who has the secret key M.

Decryption: The receiver computes: $MS = r(MQ) + MP_m = MP_m = (mMN : 1 : 0)$. Given the x-coordinate, one can devide by N and M to recover m.

Other works related to Paillier cryptosystems are in Damgård & Jurik [DJ01] and Catalano, Gennaro & Howgrave-Graham [CGH01].

g. Security on RSA-type elliptic curve cryptosystems

Many researchers have proposed and attacked RSA-type elliptic curve cryptosystems. They showed that the RSA-types of elliptic curve cryptosystems provide no significant benefits or advantages over RSA cryptosystems, even though they do resist to some known attacks on the RSA cryptosystems, if those attacks do not use integer factoring algorithms. Kurosawa, Okada & Tsujii [KOT95], Kaliski [K97] gave discussions to discourage the use of all RSA-type elliptic curve cryptosystems. Meyer & Müller [MM96] discussed that all RSA-type elliptic curve cryptosystems could be vulnerable to chosen ciphertext attacks. Håstad [H85]) proposed the low encryption multiplier/exponent attacks that is originally applied to RSA or Rabin cryptosystems when a message is encrypted with many different moduli N_i (to be sent to different users) and the encryption (public) key e is small. Håstad showed that one could solve systems of k congruence polynomials of degree e in polynomial time if $N_i > 2^{(e+1)(e+2)/4}(e + 1)^{(e+1)}$ and $k > e(e + 1)/2$. Roughly, we can have $N_i >> 2^k$. This attack is also applicable even when many public keys e_i are used instead of one, but $e = \max\{e_i\}$ satisfies the above conditions. For elliptic curve cryptosystems, this attack is called low multiplier attack (on public key). Koyama & Kuwakado [KK94a] showed that if $e \geq 5$ and $N_s = \min(N_i) \geq 2^{511}$, then KMOV and Demytko's cryptosystems are secure against the Håstad attacks. In case of KMOV scheme against Håstad attack (low multiplier attack) in broadcast applications, instead of solving congruence polynomials, one must deals with congruence rational functions that could be transferred to polynomials of bigger degree,

Bleichenbacher attack [B97] does not depend on the encryption key (public key) e, and is based on this algorithm to solve the system of equations: $b \equiv m_y^2 - m_x^3 \equiv c_y^2 - c_x^3 \pmod{N}$. It is based on a work of Coppersmith who proposed an algorithm to solve polynomial equations: $f(x) \equiv g(x) \equiv 0 \pmod{N}$, where $f(x)$ is of small degree (about 2^{32}

and $g(x)$ is a rational function of degree less than that of $f(x)$, and $g(x)$ can be computed in a small number of arithmetic steps.

Wiener [W90] showed the Small decryption key (secret key) attack that one can use the theory of continued fractions to find the secret key d of the RSA cryptosystems and RSA-type elliptic curves cryptosystems. Later, Pinch [Pi95] showed that possibility if the order of secret key d is at most $N^{1/8}$.

Homomorphism attacks. Passive homomorphism attack can be applied when the encryption E and decryption D schemes implemented are homomorphic functions to addition; that is, $E(M_1 + M_2) = E(M_1) + E(M_2)$ and $D(M_1 + M_2) = D(M_1) + D(M_2)$, where messages M_1 and M_2 are embedded on the same elliptic curve. That is, from the signatures for messages M_1 and M_2, one can forge a signature for the message $(M_1 + M_2)$ and so on for any linear combination of M_1 and M_2. The passive attack using homomorphism is usually probabilistically ineffective.

There is an active attack (chosen-plaintext attack) based on homomorphism. If an attacker wants a user B to sign a message M, he will send to user B another message: $M' = M + E(uM)$ using B's public key and a random number u. User B's signature for M' is $S' = D(M') = D[M + E(uM)] = D(M) + uM$. Then the attacker will be able to forge a signature for message M to be: $S = D(M) = S' - uM$.

A hash function should be applied on the plaintext to destroy its homomorphic property if any. However, homomorphic property is not the only condition for an RSA-type cryptosystem being vulnerable under chosen message attacks. Bleichenbacher, Joye & Quisquater [BJQ97] showed that some RSA-cryptosystems, which have no homomorphic property, are also vulnerable under even better chosen-message attacks. These attacks, which use the extended Euclidean algorithm, need only one message.

Isomorphism attacks. Passive isomorphism attack: Two elliptic curves $E_1: y^2 = x^3 + a_1x + b_1$ and $E_2: y^2 = x^3 + a_2x + b_2$ over a prime finite field F_p, where $p > 3$, are isomorphic if and only if there exists an element $u \in F_p^*$ such that $a_2 = u^4a_1$ and $b_2 = u^6b_1$. The change of variables $(x, y) \rightarrow (u^2x, u^3y)$ will transform E_1 to E_2. For elliptic curves defined over a ring Z_N the isomorphic property is similar, except the condition $u \in Z_N^*$. If this situation is satisfied, for any integer d, if we have the scalar point multiplication $(c_x, c_y) = d.(m_x, m_y)$ on the elliptic curve E_1, then we have $(u^2c_x, u^3c_y) = d(u^2m_x, u^3m_y)$ on the elliptic curve E_2.

For two randomly given plaintexts (or ciphertexts), (m_x, m_y) and (m'_x, m'_y), there is a negligible probability to have an element $u \in GF(p)^*$ such that $m'_x = u^2m_x$ and $m'_y = u^3m_y$. Hence the passive attack using isomorphism is usually probabilistically ineffective.

There is an active attack (chosen-plaintext attack) based on this isomorphism. If an attacker wants a user B to sign a message $M = (m_x, m_y)$, he will send to user B another message $M' = (u^2m_x \pmod N, u^3m_y \pmod N)$, using user B's public modulus N and a random number u. User B's signature for M' is $S' = d.(u^2m_x \pmod N, u^3m_y \pmod N) = (s'_x, s'_y)$. Then the attacker will be able forge a signature for message M to be $S = d.(m_x, m_y) = (u^{-2}s'_x \pmod N, u^{-3}s'_y \pmod N)$. To resist against this attack, a hash function should be applied on the plaintext

Concealing-message problem. Unconcealable message has its ciphertext the same as the message (plaintext) itself. Blakley & Borosh [BB79] showed that there are at least 9 messages that are unconcealable (including three trivial messages 0, –1 and 1). But there was no literature showing an analoguous problem for RSA-type elliptic curve

cryptosystems for a long time until 1997. Joye, Quisquater & Takagi [JQT97] started to analyze the concealing-message problem for RSA-type elliptic curve cryptosystems. The maximum number of unconcealable messages is bound by $(e + 1)^2$, where e is the RSA-encryption key (public key). Hence the probability that a message is unconcealable for a 1024-bit RSA modulus is as small as about 10^{-299}. But this can be open for some active attacks.

2.E.3. Elliptic curve digital signature & key agreement schemes

a. Fujioka-Fujisaki-Okamoto's scheme [FFO93]

This could be considered an elliptic curve version of the ESIGN (Efficient Digital Signature) that based on Okamoto's digital signature scheme. The idea is to extend Okamoto's scheme, whose signature involves a one-variable polynomial function, to a scheme involving a two-variable rational function. Refer to the original paper for a more detailed description. The elliptic curve used in this scheme has the form C_N: $y^2 = x^3 + ax + b$ over Z_N where $N = p^2q$. The main disadvantage of this scheme is the excessive complexity of advanced calculus used to describe the algorithm. It makes the scheme less desirable for typical technical people.

The security of this scheme depends on the difficulty of factoring $N = p^2q$. There is no known attack for this scheme so far. We did not know whether factoring $N = p^2q$ is as difficult as factoring $N = pq$. Note that the quadratic version of Okamoto's scheme was broken by Brickell & DeLaurentis [BD85] and this attack was generalized by Girault, Toffin & Vallée [GTV88].

b. An analogue of McCurley's scheme (McCurley [Mc88], Boyd & Smith [BS95])

McCurley's key agreement scheme ([Mc88]) is an enhanced variation of the Diffie-Hellman scheme. In order to break this scheme, one needs to break the ordinary Diffie-Hellman scheme and also factor big numbers. Boyd & Smith [BS95] proposed an analogue of McCurley's scheme using an elliptic curve over a ring Z_N. Then the procedure is analogous to the elliptic curve Diffie-Hellman key exchange scheme discussed earlier.

The scheme is still secure if an attacker can factor N or solve the ECDHP in the groups $E(F_p)$ and $E(F_q)$ but cannot do both. The authors also proved that if there exists an algorithm to solve the ECDHP over a ring Z_N, then it can be a feasible algorithm for factoring the modulus N.

c. Sakazaki-Okamoto-Mambo ID-based key distribution scheme

Okamoto [O88] proposed an ID-based key distribution system whose security depends on the Integer Factoring Problem as in the RSA cryprosytems. The drawback of this scheme is that it cannot be constructed on an elliptic curve over a ring Z_N in a straightforward way because the point corresponding to a user's identity may not be a point on the elliptic curve.

Mambo, Okamato & Sakazaki [MOS99] gave a solution to the above problem in order to construct it over a ring Z_N. The elliptic curve $E_N(a, b)$: $y^2 = x^3 + ax + b$, where $N = pq$, the product of two large primes, as in the RSA cryptosystems. The Center has private key consisting of p, q and k, where k = lcm $(\#E_p(a, b), (\#E_q(a, b))$. Let P be the point on $E_N(a, b)$ of order k. Given N, a, b and P, computing k is assumed to be intractable without knowledge of the prime factors p and q.

Step 1. Issuing private key to a user A whose public identifying information is ID_A.

Let $i_A = h(ID_A)$ be user A's hashed identity. Suppose that gcd(i_A, k) =1. The Center computes: $d_A = i_A^{-1} \pmod k$ and $S_A = -d_A P$ ($= -i_A^{-1} P$). The Center transmits the pair (i_A, S_A) to user A over a secure channel. Then the user A's private key is S_A and public key is i_A. It follows that $i_A S_A = -P$.

Step 2. Key exchange scheme: First, Alice chooses a random number r_A in the interval [1, $N-1$] and computes the point on the elliptic curve $E_N(a, b)$: $C_A = S_A + r_A i_B P$, and sends it to Bob. Similarly, Bob chooses a random number $r_B \in [1, N-1]$ and sends to Alice the point: $C_B = S_B + r_B i_A P$. Alice computes the point $K_{AB} = r_A(i_B C_B + P)$ and Bob the point $K_{BA} = r_B(i_A C_A + P)$. Then the share secret point is: $K = K_{AB} = K_{BA} = r_A r_B i_A i_B P$.

The above scheme can work also on the ring Z_N itself.

Qu, Stinson & Vanstone [QSV01] discussed that the scheme has multiplicative homomorphism-like properties; hence it could be vulnerable to homomorphism attacks: $S_x = -yS_{xy}$ and $S_y = -xS_{xy}$.

Suppose that gcd(x, y) = 1, $S_x = -x^{-1}P$ *and* $S_y = -y^{-1}P$. *Then* $S_{xy} = k_1 S_x + k_2 S_y$, *where* $k_1 x + k_2 y = 1$.

The attack: Suppose that one knows enough public keys I_i and secret key S_i (by interacting with the Center) to construct a database $D = \{x, S_x\}$ for small primes x, using the above lemma. Given a public key I that can be factored into primes in the database D, he/she can compute the private key S_I of that public key I.

2.F. Advantage features of elliptic curve cryptosystems

All the users can use the same underlying finite field that can be selected to optimize the finite field arithmetic. Thus it requires the same hardware to perform such arithmetic. Users still can select a different elliptic curve or can change to other elliptic curves at any time for security reasons.

All the known attacks so far can reveal a single private key at a time. The same effort has to be repeated for other private key. This is true assuming that each user employs a different elliptic curve. There is a security risk if multiple users employing the same elliptic curve and the same base point. It will take only about $(k)^{1/2}$ times as long to reveal k single private keys on the same elliptic curves.

ECC has higher cryptographic security strength with smaller in key sizes and signature sizes in comparison to other cryptosystems, RSA or DSA. An estimation by Odlyzko was presented in the table 3.1.

DSA/RSA key size	Elliptic curve key size	Time to break (MIPS Years)
512 bits	106 bits	Insecure
768 bits	132 bits	$\approx 10^8$ (not recommended)
1024 bits	160 bits	$\approx 10^{11}$
2048 bits	210 bits	$\approx 10^{20}$
2500 bits	239 bits	$\approx 10^{23}$
21000 bits	600 bits	$\approx 10^{78}$

Table 3.1. Key sizes for comparable security of DSA/RSA and ECC

Remark: A MIPS machine can perform a million microprocessor instructions per second. In cryptography literature, it is usually estimated (optimistically) that a machine rated at 1 MIPS can perform roughly 40,000 elliptic curve additions per second. A 1-MIPS year (MY) is equivalent to the computing power of a MIPS computer utilized for one year.

Computational efficiencies of ECC: faster speed in implementation and execution. Although the elliptic curve operations (such as additions or scalar point multiplications) require more arithmetic operations than modular multiplications in modulo groups, but, if the same security level is required, we will work on a much smaller size of finite fields for elliptic curves than the size of modulo groups. As a result, it is estimated that elliptic curve operations are about 8 to 10 times faster than modular multiplication in a modulo group whose order is of the same security level. There are subsequent advantages from this approach: better storage efficiencies and bandwidth savings, smaller certificates and codes, which then require smaller memory, and low cost of implementation: integrated circuit (IC) chips of smaller number of gates and lower power consumption.

The uses of elliptic curve cryptosystems will have the most benefit in applications in highly contrained and low resources environments, where bandwidth, memory, power and processing capacity are limited, such as smart cards, wireless communication devices and handheld/mobile electronics devices and computers... Hence elliptic curve cryptosystems will be the most important and efficient cryptographic technology for the next generation products of the Internet, banking, electronic commerce systems and many other security solutions. For latest work on selecting appropriate key sizes for a cryptosystem, refer to Lenstra & Verheul [LV00].

Chapter 3 – **Attacks on Elliptic Curve Cryptosystems**

We will briefly scan many well-known cryptographic attacks or algorithms on the ECDLP, the security core of elliptic curve cryptosystems. This is an active research area which will provide new algorithms on cryptographic attacks as well as counter-attacks toward the ECDLP.

3.A. Running time of algorithms

For the running time, one can count on both the number of bit operations and the number of group operations such as elliptic curve additions or scalar point multiplications or other finite field arithmetic operations…

For any integer x, the number of bits $b(x)$ of x is $[\log_2 x] + 1 = [(\log x)/(\log 2)] + 1$. Then we can write: $b(x) = O(\log x)$.

Recall the usual notation of running time function L that is usually written as a function of variable $\log x$: $L(x, c, \alpha) = O\{\exp[(c + o(1))(\log x)^{\alpha} (\log \log x)^{1-\alpha}]\}$, where α is a real number, $0 \le \alpha \le 1$ and $o(1)$ is a number that approaches 0 as x increases to infinity.

When $\alpha = 0$, $L(x, c, 0) = O((\log x)^{c'})$, then the running time is polynomial in $(\log x)$.

When $\alpha = 1$, $L(x, c, 1) = O[\exp(c' \log x)]$, then the running time is fully exponential in $(\log x)$. If written in terms of x, it is $L(x, c, 1) = O(x^{c'})$.

Otherwise, $0 < \alpha < 1$, the running time is sub-exponential in $(\log x)$. Fixing x and c, and for α in the interval [0,1], the smaller value of α will give the quicker running time.

One may distinguish two types of algorithms: General-purpose algorithm attempts to solve general problems; hence its running time depends on the size of the input. It is independent of the underlying group representation. Special-purpose algorithm attempts

to solve a particular problem; hence its running time depends not only on the size of the input but also on special features of the input or the underlying group representation.

3.B. Algorithms on the discrete logarithm problem

We will give brief discussions of most practical algorithms known in literature to find the discrete logarithm m in the equality $y = g^m$.

Shanks' Baby-step-Giant-step

This algorithm works on any group. It requires pre-computing a look-up table of powers of the base element g. Let $h = \lfloor n^{1/2} \rfloor$. We pre-compute two lists of data: $\{g^0, g^h, g^{2h}, \ldots, g^{h^2}\}$ and $\{y, yg^{-1}, \ldots, yg^{-h}\}$. Compare them to check if there is a match: $g^{uh} = yg^{-t}$, or $y = g^{uh+t}$. Then $m = \log_g y = uh + t$, for $0 \le u$ and $t \le h$.

This algorithm takes fully exponential running time in length of the largest prime factor of the order n of the group. Particularly, it is about $n^{1/2}$ steps (or group operations.) By its running time, this method is also named a square root method. It also requires memory size of $n^{1/2}$. It is a time-memory trade-off version of exhaustive search.

To resist this attack, we should have the order n of the group divisible by a big prime p, say $p \ge 10^{40} \approx 2^{132}$.

Pohlig-Hellman's method ([HP78])

This method (also referred as Silver-Pohlig-Hellman's method) reduces the problem to a determination of m modulo p_i, each of the primes p_i in the prime factorization of n, the order of the group. Then we use the Chinese Remainder theorem to recover m.

Let $n = p_1^{e_1} \ldots p_s^{e_s}$ be the order of g. Let p be any prime in the set $\{p_1, \ldots, p_s\}$. Then $z = m \pmod{p^e} = a_0 + a_1 p + \ldots + a_{e-1} p^{e-1}$, where $0 \le a_j \le p - 1$, for $0 \le j \le e - 1$. We can write $y^{n/p} = (g^{n/p})^m = (g^{n/p})^z = (g^{n/p})^{a_0}$, since $g^{n/p}$ has order p. Now, a_0 is the discrete logarithm of $y^{n/p}$ to the base $g^{n/p}$. For a_1, we write:

$$(yg^{-a_0})^{n/p^2} = g^{(m-a_0)n/p^2} = (g^{n/p})^{(m-a_0)/p} = (g^{n/p})^{a_1}.$$

Each term a_j now is a discrete logarithm to the base element $g^{n/p}$. Hence it reduces from a difficult DLP to many easier baby-DLPs. Each baby-DLP can be solved using other algorithms.

Its running time, $O(\Sigma e_i(\log n + p_i))$ group multiplications, depends mostly on the largest prime factor. Hence it works efficiently only when n is a smooth number, that is, all primes p_i are small. Therefore, in order to resist Pohlig-Hellman's algorithm, n should be divisible by a large prime number ($> 2^{80}$), or indeed, n must be prime $> 2^{80}$ for the maximum security possible.

Pollard's rho-method and lambda-method

This method is a randomized version of Shanks' Baby-step-Giant-step algorithm, and it requires no significant storage of pre-computations.

The algorithm was described in Pollard [P78] later also explained in much cryptography literature. In short, we partition the group into 3 subsets, and then perform the following recursive search: $x_{i+1} = x_i u$, where u is either x_i, g or y depending on which subset x_i belongs to. The search is completed until we find a value j such that $x_j = x_{2j}$.

By its running time, this method is also named a square root method. This algorithm also works on any group, and it takes about $(\pi n/2)^{1/2}/R$ steps (group operations) if R microprocessors are used in parallel. Refer to van Oorschot & Wiener [OW94] and [OW99]. When $n > 2^{160}$, the DLP is still infeasible.

Pollard's "lambda-method for catching kangaroos" is applicable when the result is known to be in a certain range. If the length of that range is w, then the running time is about $w^{1/2}$.

The above algorithms, as Shanks' Baby-step-Giant-step, Pollard-rho and Pohlig-Hellman, are called generic algorithms. The algorithms can work on any group and require no special group structure except that each element in the group has a unique representation.

Shoup [Sh97] showed that the lower bounds of running time for generic methods to solve DLP are proved that match the known upper bounds, about $O((n)^{1/2})$, under some assumptions. That is, in order to improve the attack efficiently, one must know more about the structure of the group. There was also a method proposed by R. Silverman & Stapleton [SS97], to solve multiple discrete logarithms: $\log_g y_1, \ldots, \log_g y_M$. This method was originally to attack the ECDLP.

Index-calculus algorithm

First we try to select an appropriate fixed subset, called the "factor base" B, of small primes g_i of the group G, such that most elements in Z_p^* can be represented as products of such primes.

We hope to be able to find the discrete logarithms $\log_g g_i$ of elements g_i in the factor base B to the base point g, by setting up a system of (a large enough number of) linear equations of the form $k = \sum_{g_i \in B} a_i (\log_g g_i) \bmod (p-1)$, where $g^k = \prod_{g_i \in B} g_i^{a_i}$.

If we can represent the given element y as a factorization over elements g_i in the factor base and the base element g: $yg^k = \prod_{g_i \in B} g_i^{b_i}$, for some random integer k, then we can solve our DLP for $x = \log_g y = \left(\sum_{g_i \in B} b_i (\log_g g_i) - k \right) \bmod (p-1)$.

This algorithm has a sub-exponential running time:

$$L(p, c, \tfrac{1}{2}) = \exp[(c + o(1))(\log p)^{1/2}(\log \log p)^{1/2}].$$

It is the fastest general-purpose algorithm for DLP.

The original (or classical) works on index-calculus methods are due to many researchers such as Maurice Kraitchik, A. E. Western & J. C. P. Miller, Adelman, Ralph C. Merkel, Pollard, Hellman & Justin M. Reyneri and Blake, Fuji-Hara, Mullin & Vanstone.

There are two sieving methods, which are under current active research.

Number field sieve algorithm

Lenstra, Lenstra, Manasse & Pollard [LLMP90] developed the Number Field Sieve method for factoring numbers (originally, of the form $n = r^e \pm s$, for r and s small.) Its running time is known heuristically as

$L(n, c, 1/3) = \exp[(c + o(1))(\log n)^{1/3}(\log \log n)^{2/3}]$, where $c = 2(2/3)^{2/3} \approx 1.526...$

It is independent of the size of factors of n. A general version of this algorithm has the value $c = (64/9)^{1/3} \approx 1.923\ldots$

Refer to the works of Coppersmith, Odlyzko & Schroeppel [COS86] the case of finite fields of characteristics 2, $c \approx 1.587$. Refer to Gordon [G93] and Schirokauer [Sc93] on the case of prime finite field F_p, $c \approx 1.923...$ For finite field $GF(p^m)$, when $m < (\log p)^{1/2}$, or $\log p \geq m^2$, the running time is heuristically sub-exponential:

$$L(p^m, c, 1/3) = \exp[(c + o(1))(\log p^m)^{1/3}(\log \log p^m)^{2/3}], \text{ for } c \approx 1.923...$$

Function field sieve algorithm (Work of Adleman, Huang, Coppersmith, Semaev…)

The works of Coppersmith and others are considered a special case of function field sieve algorithm. For finite field $GF(p^m)$, when $m > (\log p)^2$, or $\log p \leq m^{1/2}$, it has a sub-exponential running time: $L(p^m, c, 1/3) = \exp[(c + o(1))(\log p^m)^{1/3}(\log \log p^m)^{2/3}]$, for $c \approx 1.526...$

3.C. Algorithms on the elliptic curve discrete logarithm problem (ECDLP)

There is no known attack of sub-exponential time for this problem. First, in order to avoid exhaustive search, the order N of the group should be divisible by a prime $n > 2^{80}$. To avoid the square-root attacks, n should be greater than 2^{160}. For convenience, we may now assume the base point P in the ECDLP has order n.

Shanks' Baby-step-Giant-step

It takes fully exponential time in length of the largest prime factor of the order N of the elliptic curve. To resist against this attack, we should have the order divisible by a big prime n, say $n > 2^{160}$. This is one of the fastest generic algorithms for ECDLP on non-supersingular elliptic curves. It is also claimed that the best general-purpose algorithm for ECDLP is the combination of Shanks' Baby-step-Giant-step method and Pohlig-Hellman method discussed below.

Pollard's ρ-method

It takes about $(\pi n/2)^{1/2}/R$ steps (i.e., elliptic curve additions), if R microprocessors are used in parallel. Refer to van Oorschot & Wiener [OW94] and [OW99]. This is known as one of the best generic algorithms for ECDLP. It is also claimed that the best general-purpose algorithm for ECDLP is the combination of Pollard's ρ-method and the Pohlig-Hellman method. A well-known cryptographic fact is that exhaustive search through a k-bit symmetric key cipher takes about the same time as the Pollard ρ-algorithm applied to an elliptic curve cryptosystem having a $2k$-bit n. Currently, the standard n is required to be greater than 2^{160}.

Pohlig-Hellman's method

This algorithm works on any group by exploiting the subgroup structure. The idea is to determine elliptic curve discrete logarithm m in the ECDLP by determining $m \pmod{p_i}$ for all primes p_i in the prime factorization of the order n of the base point P. Then we use the Chinese remainder theorem to recover m. This algorithm works efficiently if the primes p_i are small. That is, n is a smooth number. Therefore, in order to resist the Pohlig-Hellman's algorithm, n should be divisible by a large prime number ($> 2^{160}$) and moreover, to attain the maximum security level possible, n must be prime.

In summary, no generic algorithm on ECDLP can perform substantially better than the Pohlig-Hellman algorithm combined with either Shanks' Baby-step-Giant-step

or Pollard's ρ-method. Currently, there is no other breakthrough improvement on implementation of the known attacks or of new attacks.

Method of solving multiple elliptic curve discrete logarithms (R. Silverman & Stapleton [SS97])

The method was proposed to find multiple elliptic curve discrete logarithms: $\log_P Q_1$, $\log_P Q_2$,…, $\log_P Q_M$. The idea is to apply the parallelized Pollard-rho method M times to find each $\log_P Q_i$. In each search, one may encounter two types of collisions: ordinary collisions within itself and special collisions with other searches. If the running time to find a $\log_P Q_i$ is T, then the running time to find all M discrete logarithms $\log_P Q_i$'s is $TK^{1/2}$. Hence if multiple users are using the same elliptic curve and same base point, there is a security risk, since it takes only about $K^{1/2}$ times to recover K secret keys on the same elliptic curve. Therefore, we should choose parameters such that solving each single discrete logarithm is infeasible by itself.

Index-calculus method

Over finite fields where the DLP is defined, there is another "additional structure" beyond the "multiplicative structure." The index-calculus methods take advantage of this extra structure. It is generally believed that it is much more complicated, or even impossible, to apply index-calculus method to elliptic curves. In ECDLP, the group of points has no extra structure other than the basic operation: addition of two points. A similar approach as in DLP, by choosing a "factor base" B, could not work for $E(F_q)$. The questions are: How to create a factor base for an elliptic curve? And may there be a method without requiring a factor base of elliptic curve points?

Flassenberg & Paulus [FP97] discussed that the sieving methods are still not efficient on ECDLP yet. Miller [Mi98] discussed "lifting" points on $E(GF(p^n)$ to points on an elliptic curve $\tilde{E}(\boldsymbol{Q})$ where $\boldsymbol{Q}$ is the rational field. That is, given $P \in E(GF(p^n)$, find an elliptic curve $\tilde{E}(\boldsymbol{Q})$ and a point $Q \in \tilde{E}(\boldsymbol{Q})$ such that $Q \equiv P \pmod{p}$. The natural candidate for a factor base is a set of points of small height on $\tilde{E}(\boldsymbol{Q})$. The height of an elliptic curve point that is defined as the number of bits in the numerator and denominator of the x-coordinate of that point. But these points are too sparse to generate all points on the elliptic curve by scalar point multiplications. In order to have such a lifting with probability c, the points need to have a height of at least 2^{cp}, which is impossible. Even when such a base exists, it is still a very difficult problem to find an efficient method for the lifting. Recently, Silverman & Suzuki ([S99],[SS99]) gave a more detailed proof to confirm the impossibility of index calculus method for the ECDLP. The main reason is that a factor base for the ECDLP is exponentially bigger than a DLP factor base.

In summary, the ECDLP is considered more difficult than the DLP for DSA and more difficult than the IFP (Integer Factoring Problem) on which RSA cryptosystems are based.

3.D. Application of Weil pairing and MOV reduction attack

The Weil pairing can be used to embed an elliptic curve $E(F_q)$ into the multiplicative group of the finite field $GF(q^k)$ for some positive integer k. This helps to reduce the elliptic curve discrete logarithm problem (ECDLP) on the curve $E(F_q)$ to the ordinary discrete logarithm problem in the multiplicative group $GF(q^k)^* = GF(q^k)\backslash\{0\}$. This method is called MOV reduction (proposed by Menezes, Okamoto & Vanstone

[MOV93]). We need an assumption that gcd(#($E(F_q)$, q) = 1, and it works on both cases, either $q = p > 3$, or $q = 2^m$. The supersingular elliptic curves automatically satisfy the assumption that gcd(#$E(F_q)$, q) = 1, since if $q = p$, then $E(F_q) = p + 1$, while if $q = 2^m$ then #$E(F_q)$ must be an odd number.

Let P be the base point of order n used in the ECDLP. Let Q be a fixed $GF(q^k)$-point of order n that is not in the subgroup $\langle P \rangle$, generated by point P. The mapping defined by Weil pairing, $P \mapsto e_n(P, Q)$, is the embedding of subgroup $\langle P \rangle$ into the multiplicative subgroup $GF(q^k)^*$ of the finite field $GF(q^k)$. The necessary condition for this embedding is obviously that:

(**a**) *The order n of point P must divide* $(q^k - 1)$.

Then the finite field $GF(q^k)$ contains all n n-th roots of unity. We need one more condition for MOV working on the finite field $GF(q^k)$:

(**b**) *The elliptic curve* $E(GF(p^k)$ *contains* n^2 *points of prime order n.* Hence we have $E[n](F_q) \subseteq E(GF(p^k))$.

It was shown that statement (**b**) implies statement (**a**) and the proof did not require either condition $n \mid \#E(F_q)$ or $n \nmid (q - 1)$.

Balasubramanian & Koblitz [BK98] showed that the statement (**a**) implies statement (**b**), if two conditions $n \mid \#E(F_q)$ and $n \nmid (q - 1)$ are satisfied. Indeed, since $n \mid \#E(F_q)$, the elliptic curve $E(F_q)$ contains a point $P \neq O$ of order n. The condition $n \mid (q^k - 1)$ implies that gcd(n, q) = 1. Hence the elliptic curve $E(GF(p^s)$ contains n^2 points of order n, for some positive integer s. Using Frobenius mapping and the critical condition $n \nmid (q - 1)$, they proved that, in fact, s must be k. We have (**b**). q.e.d.

Theorem (Balasubramanian & Koblitz [BK98]): *Let n be a prime such that* $n \mid \#E(F_q)$ *and* $n \nmid (q - 1)$. *Then the elliptic curve* $E(GF(q^k))$ *contains* n^2 *points of order n if and only if* $n \mid (q^k - 1)$, *for some positive integer k.*

That is, the necessary condition $n \mid (q^k - 1)$ is in fact also sufficient for MOV reduction, provided a crucial assumption that $n \nmid (q - 1)$. Otherwise the result is wrong even if one assumes that $n^2 \mid \#E(F_q)$.

The running time for DLP in this situation will be sub-exponential in $\log(q^k)$, by index calculus method, $L(q^k, c, 1/3) = \exp[(c + o(1))(\log q^k)^{1/3}(\log \log q^k)^{2/3}]$. In fact, the algorithm is not proved for the case q prime and $k > 1$, but cryptography researchers would prefer to assume so "optimistically." Refer to Gordon [G93] for the case $k = 1$.

Hence the MOV reduction is significant only when k is small enough. Particularly, if we have $k \geq (\log q)^2$, its running time will be greater than

$$\exp[(c')(\log q^{(\log q)^2})^{1/3}(\log\log q^{(\log q)^2})^{2/3}] =$$
$$= \exp[(c')\log q\ (3\log\log q)^{2/3}] > \exp[(c')\log q].$$

This running time is fully exponential in $(\log q)$ and hence there is no significant advance in the MOV reduction. When $k = \log q$, we can estimate that
$[(\log q^{\log q})^{1/3}(\log\log q^{\log q})^{2/3}] = (\log q)^{2/3}.(2\log\log q)^{2/3} < (\log q)^{2/3}(\log q)^{1/3} = \log q$.
Then the running time is sub-exponential. Hence the borderline value for k could be $\log q < k < (\log q)^2$. One could choose roughly $k_0 = (\log q)^2/[s(\log\log q)^2]$, for a value s such that $0 < s < q$, for k to be on the borderline of exponential and sub-exponential time.

We consider supersingular and non-supersingular elliptic curves separately.

When E is supersingular, the MOV reduction needs the extension field to be of degree $k \leq 6$ only. Let $\#E(F_q) = q + 1 - t$. Then the values of k are computed as follows:

t^2	0	q	$2q$	$3q$	$4q$
k	2	3	4	6	1

Table 3.2. Values of k versus t^2

Hence the MOV method is successful on attacking the ECDLP on supersingular elliptic curves. That is the reason that one generally prefers non-supersingular elliptic curves in applied cryptography.

When E is non-supersingular, to resist the MOV attack, we must have $k > (\log q)^2$ (with a very high margin of security) and this can be done simply. Particularly, the order n of base point P must be checked to satisfy the MOV condition, that is: $n \nmid (q^k - 1)$ for all $k \le (\log q)^2$. In practice, we do not need value of k to be near $(\log q)^2$ (as we also observed above), but $k = 20$ is sufficient. In the future, we should replace this value by a number b such that the DLP on $GF(q^b)$ is intractable with the up-to-date technology capability.

Assume that n is divisible by a large prime u (or if n is a prime itself, then $u = n$) in order to resist the Pohlig-Hellman attack. To ensure the condition: $E[u] \not\subset E(GF(q^t))$ for each t, $1 \le t \le k$, it is sufficient to check either $u^2 \nmid \#E(GF(q^t))$ or $n \nmid (q^t - 1)$.

With such requirements, the ECDLP on non-supersingular elliptic curves has no known sub-exponential time algorithm. The best-known algorithms (which still run at exponential speed) are Shanks' Baby-step-Giant-step and MOV reduction followed by a number field sieve (Koblitz, 1991), or combination of Pollard-ρ and Pohlig-Hellman methods. Refer to [HMV93]. Currently, there is no other breakthrough improvement on implementation of the known attacks or of new attacks.

Supersingular elliptic curves $\#E(F_q)$	
Over F_p, $p > 3$	E: $y^2 = x^3 + ax + b$ when $\#E(F_p) = p + 1$
Over $GF(2^m)$	E: $y^2 + cy = x^3 + ax + b$ whose order will be one of the followings: $2^m + 1$, $2^m + 1 \pm 2^{m/2}$, $2^m + 1 \pm 2^{(m+1)/2}$ or $2^m + 1 \pm 2^{(m+2)/2}$.

Table 3.3. A checklist of orders of supersingular elliptic curves that should be avoided

Chao, Tanada & Tsujii [CTT94] discussed that the minimum k satisfies $n \mid (q^k - 1)$, or $q^k \equiv 1 \pmod{n}$, must be a factor of $\phi(n)$, that is equal to $(n - 1)$ in the case n is prime. Hence in order to check that $k > B$, some lower boundary, one may only check that $\phi(n)$ is B-nonsmooth, (i.e., $\phi(n)$ has no prime factors less than B), instead of factoring $\phi(n)$ completely.

3.E. SSA attack (Smart-Satoh-Araki attack)

The SSA attack is to solve the ECDLP on non-supersingular elliptic curves of trace $t = 1$ or $\#E(F_p) = \#(F_p)$, where $p > 3$. They are called anomalous curves. The idea is to reduce the ECDLP to a simpler equation in a p-adic field. There are a few independent works on this problem by Smart, Samaev and Araki & Satoh. The generalized work is done recently by Rück. The SSA attack cannot apply to other cases since its proof uses the essential identity: $\#E(F_p) = \#(F_p) = p$. In order to resist this attack, the SSA condition (or anomalous condition) should be satisfied: $\#E(F_p) \ne \#(F_p) = p$.

One should note that the anomalous binary curves (ABC's) or "Koblitz curves," which are defined over finite fields $GF(2^m)$, are not susceptible to the Smart-Satoh-Araki

attack. It was known that trace-1 elliptic curve case was mentioned earlier at the West Coast Number Theory conference, 1996, as a "well known fact" and hence unpublished.

Araki & Satoh's work ([AS98]) is to solve the ECDLP on non-supersingular elliptic curves of trace $t = 1$ or $\#E(F_p) = \#(F_p)$. The method is based on an elliptic curve version of Fermat quotient that is defined as $L_p(a) = (a^{p-1} - 1)/p \pmod p \in F_p$, for a such that $\gcd(a, p) = 1$. It is well-defined because of Fermat's Little theorem, $a^{p-1} - 1 \equiv 0 \pmod p$.

The detailed proof is out of context. It utilized the concept of formal group, associated with an elliptic curve, and its properties. This algorithm has a linear running time $O((\log p)^3)$ to solve the ECDLP on an anomalous elliptic curve. It also works on non-prime finite fields F_q, where $q = p^r$, where $p > 3$.

Smart's work ([Sm99]): claimed that this attack works just on prime finite fields. The idea is to reduce the ECDLP to a simpler equation in a p-adic field. We consider the ECDLP as usual, $Q = m \cdot P$, on an elliptic curve, and neither point has order two. We then lift these points to points $\overline{P}$ and $\overline{Q}$ on the same elliptic curve but over the p-adic field $\mathbf{Q}_p$ by $\overline{P} = (x, y)$ where x is the x-coordinate of P and y is computed via Hensel's lemma, and similarly for the point $\overline{Q}$. Then we have: $\overline{Q} - m\overline{P} = \overline{R} \in E_1(\mathbf{Q}_p)$, where $E_n(\mathbf{Q}_p) = \{P \in E(\mathbf{Q}_p)\colon v_p(x(P)) \le -2n\} \cup \{\mathrm{O}\}$, as a subgroup of the elliptic curve $E(K)$, and v_p is the discrete valuation in $\mathbf{Q}_p$. Then we have two following equivalence relations:

$$E_0(\mathbf{Q}_p)/\, E_1(\mathbf{Q}_p) \cong E(F_p) \text{ and } E_n(\mathbf{Q}_p)/E_{n+1}(\mathbf{Q}_p) \cong E_1(\mathbf{Q}_p)/\, E_2(\mathbf{Q}_p) \cong F_p, \text{ for } n \ge 1.$$

If the elliptic curve satisfies the condition: $\#E(F_p) = \#(F_p) = p$, we can multiply the above equation by p: $p\overline{Q} - m \cdot p\overline{P} = p\overline{R} \in E_2(\mathbf{Q}_p)$. Taking the p-adic elliptic logarithm Ξ_p (which is easy to compute but its context is not able to be described yet within this document), we have: $\Xi_p(\, p\overline{Q} - m.\, p\overline{P}\,) = \Xi_p(\, p\overline{R}\,) \equiv 0 \pmod{p^2}$. This reduces the ECDLP to a single linear equation of one unknown over the p-adic field $\mathbf{Q}_p$. Its solution is $m = \Xi_p(\, p\overline{Q}\,) \,/\, \Xi_p(\, p\overline{P}\,) \pmod p$.

This attack has a linear running time $O(\log p)$, since the only non-trivial computation needed to be performed is to compute $p\overline{Q}$ and $p\overline{P}$. Both computations need a number of $(\log p)$ group operations on an elliptic curve.

Semaev [Se98] showed a more general result on the elliptic curve discrete logarithm problem in a subgroup of order p of an elliptic curve $E(F_q)$, where $q = p^k$. Semaev showed how to construct an isomorphism from such a subgroup to an additive subgroup of some finite field $GF(p^k)$. Then the problem can be solved in polynomial running time, $O(\log p)$ field operations. As an immediate result, the ECDLP in anomalous elliptic curves over prime finite fields then can be solved easily in polynomial running time. Rück [R97] generalized the result of Semaev [Se98] for more general curves.

3.F. Differential and power attacks

More rigorous proofs for those estimations are still needed in order to evaluate and compare the security of various cryptosystems. More powerful attacks are expected for the years to come when elliptic curve cryptosystems are more extensively studied and widely implemented. There are a few estimations on the security of elliptic curve cryptosystems, in comparisons with other cryptosystems, scattered in cryptography

literature. For example, refer to the table 3.4 for the case $n > 2^{160}$, where Vaudenay's attack on DSA (or DLP) was blocked in ECDSA (or ECDLP).

(Length in bits)	System parameter	Public key °	Private key	Signature size	Encrypted 100-bit message
RSA	N/A	1024+64	2048	1024	1024
DSA	2208	1024	160	320	depend
ECC	481	161	160	320	321

Table 3.4. Comparisons of the cryptosystems RSA, DSA and ECC
(°: For time to break of about 10^{11} MIPS years)

The software attack is to use an exhaustive search for a solution of an ECDLP.

The hardware attack is to build special-purpose hardware for a parallel search by Pollard-ρ method that is considered one of the best algorithms so far, or used in combination with other methods or even by new algorithms hopefully to be developed. Both software and hardware attacks are still infeasible for elliptic curve cryptosystems with at least $n > 2^{160}$, unless there will be significant breakthroughs in hardware designs, computational or cryptanalytic algorithms.

Side-channel attacks: The timing and differential power attacks belong to the family of attacks, called "side-channel" attacks, which devise to exploit the leakage of information from implementations of cryptosystems.

Timing attacks, which are proposed by Kocher [Ko96] are based on repeatedly measuring the exact execution times of modular exponentiation operations. Kocher also proposed attacks using differential power analyses based on the power assumption of cryptographic devices. This later type is usually referred as Simple Power Attacks (SPA). To defend against such attacks, in general, one should uniformize or homogenize the computations to make running time and power independent of key bits or randomize inputs and key bits. Particularly for elliptic curve cryptosystems, one should use Montgomery's method for point scalar multiplication or add dummy operation to homogenize the point adding operations.

Differential fault & power analyses are one of important issues recently for cryptographic attacks, particularly against ECC. Refer to Biehl, Meyer & Müller [BMM00], Coron [C99], Aigner & Oswald [AO01] and Joye & Tymen [JT01] for discussions on both attacks and counter-attacks. The differential fault attack induces computational errors to the device and deduce key bits from the leaked information by the faulty results. To defend againt DFA attack, one should check the consistency of the computational results. For example, for elliptic curve cryptosystems, verifying the resulting point being on the elliptic curve is implicit consistency relation which should be used at all time.

The differential power attack attack applies statistical tests to intermediate results in order to detect correlations between plaintext and ciphertext. To defend against DPA attack, one should decorrelate the intermediate results, key bits, plaintexts and ciphertexts by randomization. Particularly for elliptic curve cryptosystems, we can also use the randomized projective coordinates.

Later works, such as Bellezza [Be01], also discussed many methods of counter-attacks against the side-channel attacks, especially for the elliptic curve cryptosystems, such as: moving to a random isomorphic elliptic curve, changing the field representation,

(such as using projective coordinates…) and adding a random point to the input and subtracting a suitable multiple of that point at the end. But in a general sense, designers should be aware that design and implemention of a countermeasure against one physical cryptanalysis may benefit another attack or attacks.

Chapter 4 – **Implementations of Elliptic Curve Cryptosystems**

We will discuss many technical issues in implementations of elliptic curve cryptosystems such as how to implement finite fields, elliptic curves and arithmetic operations on them. More techniques and algorithms are likely being developed to improve the efficiency of elliptic curve cryptosystems. An elliptic curve cryptosystem requires a process of setting up common system parameters. These parameters, once generated, will be used by all users within a group using that cryptosystem. Each user will have his/her own pair of private and public keys.

4.A. Implementations of finite fields

First, one has to decide which finite fields will be used for an elliptic curve cryptosystem. One must choose an appropriate finite field and its basis to represent field elements.

4.A.1. Finite fields

In order to implement finite fields, we will deal with these tasks:

Selecting the underlying finite field F_q.

Selecting a basis representation for the finite field elements,

Implementing the arithmetic on the finite field F_q.

The cost, speed and feasibility of elliptic curve cryptosystems depend on the finite field F_q, where $q = p^m$, on which it is implemented. There are usually two finite fields to work on: prime finite field $F_p = Z_p$ (i.e., $m = 1$) when p is a prime number > 3 and binary finite field $GF(2^m)$.

a. Prime finite fields F_p, where p is a large prime number > 3

The prime p should be large enough such that the ECDLP is infeasible over prime finite field F_p. The minimum threshold of the choice of p should increase as new technology and theory develops to attack the ECDLP. In fact, it is the order of an elliptic curve that is the first factor in selecting the elliptic curve. As we will discuss later, that order must be divisible by a prime > 2^{160}. It is also within a relatively small range around p, the order of F_p, as we knew already: $| p - \#E(F_p)| \le 1+ 2\sqrt{p}$.

Mersenne primes. The prime p is suggested to be a Mersenne prime for maximum security and efficient implementation. There is also general doubt that the more special internal structures are embedded in a finite field chosen for an elliptic curve cryptosystem, the more vulnerable this cryptosystem will be.

A Mersenne prime is of the form $p = 2^t - 1$, where t is, obviously, a prime. We call t a Mersenne exponent. For example, t = 2, 3, 5, 7, 17, 31, 89, 127, 521 and 607, we have $(2^t - 1)$ to be a Mersenne prime.

A Mersenne number is of the form $(2^t - 1)$, where t is not necessarily a prime. Not any prime can be a Mersenne exponent; e.g., $2^{11} - 1 = 2047 = 23 \cdot 89$ is a composite number.

Pseudo Mersenne primes. In Crandall's patent [C92], the Mersenne primes are a subset of fast class numbers, which are primes of the form $p = 2^t - C$, where C is a very small positive number in practice. They are called pseudo Mersenne primes. Then the modular reduction (modulo p) can be implemented very efficiently in this case: using only cyclic shifts and additions, with no divisions required. This advantage is also for the Fermat primes of the form $p = 2^{2^s} + 1$.

Generalized Mersenne primes. Generalized Mersenne primes are also implemented for prime finite fields. They are primes of the form $p = 2^n - 2^s - 1$.

Modular reduction method. The binary form of a number a is represented as

$$a = \underbrace{a_{nt-1}a_{nt-2}\ldots a_{(n-1)t}}_{t \text{ terms}} \cdots \underbrace{a_{it+(t-1)}\ldots a_{it+1}a_{it}}_{t \text{ terms}} \cdots \underbrace{a_{t-1}\ldots a_1 a_0}_{t \text{ terms}} = A_{n-1}\ldots A_1 A_0.$$

where each set of t bits, starting from the far right side, are grouped. Let $p = 2^t - C$. Then we have: $2^t \equiv C \pmod p$. Hence

$$A_i = (a_{it+(t-1)}\ldots a_{it+1}a_{it}) = 2^{it}\sum_{j=0}^{t-1} a_{it+j}2^j \equiv C^i \sum_{j=0}^{t-1} a_{it+j}2^j \pmod p.$$

That is, to compute $A_i \pmod p$ we just shift its bits to the positions for those of A_0, then we compute its decimal value. Adding them up, we get the modulo p of a

$$a \pmod p \equiv A_{n-1} \pmod p + \ldots + A_0 \pmod p.$$

When $C = \pm 1$, the algorithm will have only shiftings and additions/subtractions.

When $C \neq \pm 1$, the algorithm will actually have shiftings, multiplications by C, and then additions/subtractions.

b. Finite fields $GF(2^m)$ of characteristic 2 (binary finite fields)

We now discuss more on the bases of $GF(2^m)$ that is considered a vector space of dimension m over F_2. A basis is a set of elements $\{e_0, e_1,\ldots, e_{m-1}\}$ in $GF(2^m)$ such that each element $a \in GF(2^m)$ can be represented uniquely by the form $a = \sum_{i=0}^{m-1} a_i e_i$, where $a_i \in F_2$, i.e., $a_i = 0$ or 1. Then one can write: $a = (a_0, a_1,\ldots, a_{m-1})$.

There are many different bases of $GF(2^m)$. The most natural bases are, of course, polynomial bases, normal bases and optimal normal bases.

c. Extension finite fields

Extension finite fields are special cases of practical implementation. They are divided into two groups: composite finite fields of characteristic 2 are of the form $GF(2^n)^m)$ and Optimal Extension Fields (OEF) of the form $GF(2^n \pm c)^m)$ have characteristic greater than 2. We can summarize all types of finite fields, which are currently implemented, into the table 4.1.

Finite Fields					
Prime finite fields $GF(p)$, $p > 2$			Extension finite fields $GF(p^m)$		
General primes (& Mersenne)	Special form primes		char $(p) = 2$		char $(p) > 2$ OEF
	Pseudo Mersenne	Generalized Mersenne	Binary	Composite	
$GF(p)$	$GF(2^n - c)$	$GF(2^n - 2^s - 1)$	$GF(2^n)$	$GF((2^n)^m)$	$GF((2^n \pm c)^m)$

Table 4.1. Finite field classifications

4.A.2. Polynomial bases

Let $f(x) = x^m + f_{m-1}\, x^{m-1} + \ldots + f_1 x + f_0$, where $f_i \in F_2$, be an irreducible polynomial of degree m over F_2. We call $f(x)$ the reduction polynomial (or sometimes, field polynomial). Let $\alpha = x + (f(x))$ be a root of $f(x)$. Hence the set $\{\alpha^{m-1},\ldots,\alpha^2, \alpha, 1\}$ is a polynomial basis, where $\alpha^i = x^i + (f(x))$, for $0 \le i \le m-1$.

Then the elements of the finite field $GF(2^m)$ can be represented as the set of all polynomials of degree $0 \le d \le m-1$: $a(x) = a_{m-1}\, x^{m-1} + \ldots + a_1 x + a_0$, where $a_i = 0$ or 1. One can also write: $a(x) = (a_{m-1},\ldots, a_1, a_0)$. Particularly, we can write the zero element 0 = (0, 0,…, 0) and the multiplicative identity 1 = (0, 0,…, 1).

a. Irreducible polynomials

A polynomial $f(x)$ is called irreducible if we cannot write $f(x) = g(x).h(x)$, for any polynomials $g(x)$, $h(x)$ of degree strictly less than the degree of $f(x)$.

An irreducible polynomial $f(x) = x^m + f_{m-1}\, x^{m-1} + \ldots + f_1 x + f_0$ of degree m over F_2 should satisfy these necessary conditions:

The constant term $f_0 = 1$; otherwise, we can factor x out. Hence from now on, we always write the general form as: $f(x) = x^m + f_{m-1}\, x^{m-1} + \ldots + f_1 x + 1$.

There is an odd number (≥ 3) of nonzero terms; otherwise, $f(x)$ whose number of nonzero terms is even has a factor $(x + 1)$.

There must be at least one term of odd degree; otherwise, $f(x)$ of all even powers is a square of a polynomial of degree $(m/2)$.

It is easy to verify this property:

If $f(x) = x^m + f_{m-1}\, x^{m-1} + \ldots + f_1 x + 1$ is an irreducible polynomial of degree m, then so are polynomials $g(x) = f(x + 1)$ and $h(x) = x^m.f(1/x) = x^m + f_1\, x^{m-1} + \ldots + f_{m-1} x + 1$.

The compositions of $g(x)$ and $h(x)$ will give us a few more irreducible polynomials.

b. Primitive polynomials

If $f(x) = f_m\, x^m + \ldots + f_1 x + f_0$ is an irreducible polynomial of degree m over F_2 and r is a root of $f(x)$ in an extension field of F_2 (that is a finite field $GF(2^m)$), then r, r^2, r^{2^2},..., $r^{2^{m-1}}$ are all roots of $f(x)$. Indeed, if $f(r) = 0$, then for any d, we have $f_i^{2^d} = f_i$, since f_i equals either 0 or 1. Hence:

$$f(r^{2^d}) = f_{m-1} r^{2^d.(m-1)} + \cdots + f_1 r^{2^d} + f_o = (f_{m-1} r^{m-1} + \cdots + f_1 r + f_o)^{2^d} = (f(r))^{2^d} = 0.$$

Property: *All the roots of the polynomial $f(x)$ have the same multiplicative order, that is called the period (or order) of the function $f(x)$.*

PROOF: Indeed, if order of r is k, i.e., $r^k = 1$, then $k \mid (2^m - 1)$. Hence k is odd. We also have $r^{k.2^d} = (r^{2^d})^k = 1$. Therefore, the order of r^{2^d} is some integer l such that: $l \mid k$ and $r^{2^d.l} = 1$. Hence, $k \mid (2^d l)$. Since k is odd, we have $k \mid l$. Hence $k = l$. q.e.d.

The period (or order) of a polynomial $f(x)$ can also be defined as the least positive integer e such that $f(x)$ divides the polynomial $(x^e + 1)$.

Definition: *If the period of f(x) is* $(2^m - 1)$, *that is the order of the multiplicative subgroup* $GF(2^m)^* = GF(2^m)\backslash\{0\}$, *then f(x) is called a primitive polynomial*.

All roots of a primitive polynomial $f(x)$ are primitive elements of $GF(2^m)$. For example, the polynomial $h(x) = x^m + \ldots + x + 1$ divides $(x^{m+1} + 1)$. Hence its period is equal to $(m + 1)$ or less and $h(x)$ is not a primitive polynomial of $GF(2^m)$, unless $m = 2$.

A primitive polynomial can be built from a primitive element a of a finite field $GF(2^m)$ by the formula: $f(x) = (x + a)(x + a^2)(x + a^{2^2}) \cdots (x + a^{2^{m-1}})$.

If another primitive polynomial $g(x) = (x + b)(x + b^2)(x + b^{2^2}) \cdots (x + b^{2^{m-1}})$ is built from a primitive element b, such that $b \notin \{a^{2^n} \mid 0 \le n \le m-1\}$, then $g(x) \neq f(x)$.

In other words, each primitive element is a root of only one primitive polynomial of $GF(2^m)$. In fact, the set of roots of all primitive polynomials for a finite field are exactly all primitive elements in $GF(2^m)^* = GF(2^m)\backslash\{0\}$. Hence there can be many primitive polynomials for a finite field $GF(2^m)$, when $m \ge 3$. And in fact, a primitive polynomial cannot be reducible over F_2. [8]

Recall that the number of non-zero terms for a reduction polynomial must be odd. Hence the first polynomials to be considered are of 3 non-zero terms since we prefer the fewest terms in reduction polynomials. But irreducible trinomials are relatively sparse; hence the next candidates are polynomials of 5 non-zero terms or pentanomials. In practice, the most-used polynomial bases are trinomial and pentanomial bases. We can choose such a reduction polynomial $f(x)$ such that the computations modulo $f(x)$ can be performed efficiently in software and hardware implementations.

c. Trinomial basis representation

Its reduction polynomial is an irreducible trinomial of the form $T_{m,k}(x) = x^m + x^k + 1$, where $1 \le k \le m-1$. In fact, a trinomial $T_{m,k}(x) = x^m + x^k + 1$ is irreducible if and only if its reciprocal trinomial $T_{m,m-k}(x) = x^m + x^{m-k} + 1$ is irreducible. Hence, we should be interested in trinomials of the following form only: $T_{m,k}(x) = x^m + x^k + 1$, where $1 \le k \le m/2$. Such trinomials exist for certain values of m only. If they exist, we should choose the reduction polynomial with the smallest k. Such a trinomial generally will have the most efficient implementation.

d. Pentanomial basis representation

Its reduction polynomial is an irreducible pentanomial of the form

$$P(x) = x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1, \text{ where } 1 \le k_1 < k_2 < k_3 \le m-1.$$

Such pentanomials always exist for $m \ge 4$. In practice, it was recommended to use pentanomials whose coefficient triples (k_1, k_2, k_3) or (k_3, k_2, k_1) will have the first coefficient as small as possible and next coefficients are kept as small as possible after fixing the previous one or ones in the triple order. These polynomials would have more efficient computations of finite field operations.

e. The field arithmetic

Its operations are performed via modulo $f(x)$ over the finite field F_2 as follows:

The reduction modulo $f(x)$ of a polynomial $g(x)$ is just the remainder when $g(x)$ is divided by $f(x)$.

Field addition is performed component-wise by XOR-ing.

---

[8] **Number theory tip – Number of primitive polynomials**

There are a total of $\phi(2^m - 1)$ primitive elements in the multiplicative subgroup $GF(2^m)^* = GF(2^m) \backslash \{0\}$.

Property: For any integer $m > 0$, we have $m \mid \phi(2^m - 1)$, or more generally, $m \mid \phi(p^m - 1)$, for any prime p. Hence the number of primitive polynomials of a finite field $GF(2^m)$ is equal to $\phi(2^m - 1)/m$.

$(a_{m-1},\ldots, a_1, a_0) + (b_{m-1},\ldots, b_1, b_0) = (c_{m-1},\ldots, c_1, c_0)$ where $c_i = a_i \oplus b_i$.

Field multiplication: $(a_{m-1},\ldots, a_1, a_0) \cdot (b_{m-1},\ldots, b_1, b_0) = (r_{m-1},\ldots, r_1, r_0)$, where the polynomial $(r_{m-1}x^{m-1} + \ldots + r_1x + r_0)$ is the remainder when the product

$$(a_{m-1}x^{m-1} + \ldots + a_1x + a_0) \cdot (b_{m-1}x^{m-1} + \ldots + b_1x + b_0)$$

is divided by $f(x)$ over F_2.

Refer also to Schönhage [S77] and Pincin [P89] for more disccusions. Kim, E. J. Lee & P. J. Lee [KLL98] proposed a new inversion algorithm, called MAIA (Modified Almost Inverse Algorithm), which is suited especially for Optimal Extension Fields.

Squaring. In particular, the squaring operation of a polynomial $(a_{m-1}x^{m-1} + \ldots + a_1x + a_0)$ that is performed in modulo 2 is in fact a linear operation; that is,

$$(a_{m-1}x^{m-1} + \ldots + a_1x + a_0)^2 = a_{m-1}x^{2(m-1)} + a_{m-2}x^{2(m-2)} + \ldots + a_1x^2 + a_0.$$

In terms of bit strings, we write: $(a_{m-1},\ldots, a_1, a_0)^2 = (a_{2(m-1)}, 0, a_{2(m-2)}, 0,\ldots, 0, a_1, 0, a_0)$. Then we reduce the resulting polynomial by modulo $f(x)$.

We should refer to some algorithms using squaring matrices when we need to implement many squaring operations in a fixed polynomial basis.

Orlando & Paar [OP00] proposed a design for the standard basis field representation, and it is based on the transformation from squaring operation into an addition and a multiplication by a constant that depends only on the field polynomial. Let $L = \lceil m/2 \rceil$ and $K = \lfloor m/2 \rfloor$, then

$$(a_{m-1}x^{m-1} + \ldots + a_1x + a_0)^2 = a_{m-1}x^{2(m-1)} + a_{m-2}x^{2(m-2)} + \ldots + a_1x^2 + a_0.$$

$$= x^{2L}[a_{L+(K-1)}x^{2(K-1)} + a_{L+(L-2)}x^{2(L-2)} \ldots + a_{L+1}x^2 + a_L] + [a_{L-1}x^{2(L-1)} + \ldots + a_1x^2 + a_0].$$

$= A.B + C$, where $A = x^{2L} \bmod f(x)$ is a constant depending only on the field representation and A could be reduced to a polynomial of degree much less than m and $B = a_{L+(K-1)}x^{2(K-1)} + a_{L+(L-2)}x^{2(L-2)} \ldots + a_{L+1}x^2 + a_L$ and $C = a_{L-1}x^{2(L-1)} + \ldots + a_1x^2 + a_0$.

Inversion: We need to discuss also some methods of computing the inverse of a non-zero element. This operation obviously has an important role in field arithmetic. The general method is using this identity: $a^{-1} = a^{2^m-2} = (a^{2^{m-1}-1})^2$, $\forall a \neq 0$. Recall that: $a^{2^m-1} = 1$. In implementations, we can even analyze further the power exponent $(2^{m-1} - 1)$ of a to reduce our computation to a few multiplications.

Another well-known method is using the Euclidean algorithm. After finding gcd($f(x)$, $a(x)$), we can work backward the steps in the Euclidean algorithm to represent gcd($f(x)$, $a(x)$), as a linear combination of $f(x)$ and $a(x)$. This is called the *extended Euclidean algorithm*: finding polynomials $u(x)$ and $v(x)$ such that gcd($f(x)$, $a(x)$) = $f(x)\cdot u(x) + a(x) \cdot v(x)$. When gcd ($f(x)$, $a(x)$) = 1, we can write $1 \equiv a(x)\cdot v(x) \pmod{f(x)}$. In other words, the polynomial $v(x)$ is the inverse of $a(x)$, modulo $f(x)$.

The explicit algorithm and its hardware architecture can be found easily in computer engineering literature. In fact, it is generally referred to as an algorithm to compute the ratio $b(x)/a(x)$. Hence inversion is only a special case when we let $b(x) = 1$.

Quite a few methods are mentioned in cryptography literature. For each particular finite field and its chosen reduction polynomial, one method can be implemented more efficiently or conveniently than others.

O'Malley, Orman, Schroeppel & Spatscheck [OOSS95] proposed the "Almost Inverse" algorithm on the binary finite field $GF(2^{155})$ with its reduction polynomial $T(x) = x^{155} + x^{62} + 1$. In fact, this is the only irreducible trinomial of $GF(2^{155})$ (if one ignores

its reciprocal trinomial) and it is not a primitive polynomial. The authors developed the "*Almost Inverse*" algorithm from Elwyn R. Berlekamp's idea (in his book (*Algebraic coding theory,* 1968)) and the low-end GCD algorithm, which were independently discovered by Silver & Terzian, 1962 (never published) and Stein [S67].

The idea of "Almost inverse" algorithm is to compute the almost inverse of a polynomial $a(x)$ modulo $f(x)$. It will compute a polynomial $b(x)$ and an integer j (which is less than twice the degree of $f(x)$,) such that $a(x)\, b(x) \equiv x^j \pmod{f(x)}$. Then the inverse of the polynomial $a(x)$ will be the polynomial $b(x)$ divided by the term x^j.

They also suggested a few similarly well-behavior irreducible trinomials: $x^{127} + x^{63} + 1$, $x^{140} + x^{65} + 1$, $x^{182} + x^{81} + 1$, $x^{191} + x^{71} + 1$, $x^{223} + x^{91} + 1$ and $x^{255} + x^{82} + 1$.

Irreducible trinomials are rather sparse. For finite fields of degrees between 100 to 199, there are 43 fields having no irreducible trinomials.

4.A.3. Normal bases and optimal normal bases

Normal bases are not special only for finite fields of characteristic 2. In fact, they are defined for any finite field $GF(q^m)$ where q is a prime power.

A normal basis of $GF(2^m)$ over F_2 is a basis of the form $\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\}$, where $\beta \in GF(2^m)$. Such a basis always exists. Then $a = (a_0, a_1, \ldots, a_{m-1})$ will represent the element $a = a_0\beta + a_1\beta^2 + a_2\beta^{2^2} + \cdots + a_{m-1}\beta^{2^{m-1}}$. By convention, the ordering of bits in normal basis representation is different from that in polynomial basis representation. Particularly, we can write the zero element 0 = (0,0,…,0) and multiplicative identity 1 = (1,1,…,1).

The most important property of a normal basis is that the square of a field element can be computed easily and implemented efficiently on hardware by just a right 1-cyclic shift on the register. Indeed, given an element $a = (a_0, a_1, \ldots, a_{m-1})$ represented in a normal basis, we have: $a^2 = \left(\sum_{i=0}^{m-1} a_i \beta^{2^i}\right)^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} = (a_{m-1}, a_0, a_1, \ldots, a_{m-2})$. Then for any integer s, $1 \le s \le m-1$, the 2^s*-th* power of element a can be computed quickly by an right s-cyclic shift. That is, $a^{2^s} = (a_{m-s}, a_{m-s+1}, \ldots, a_0, a_1, \ldots, a_{m-s-1})$. We can verify again the relationship: $a^{2^m} = a$. Similarly, the square root of a can be computed simply by a left 1-cyclic shift: $a^{1/2} = (a_1, a_2, \ldots, a_{m-1}, a_0)$.

Unfortunately, multiplication in a normal basis is more complicated.

a. The field arithmetic

Field addition is performed component-wise by XOR-ing as on a polynomial basis. Field multiplication: $(a_0, a_1, \ldots, a_{m-1}) \cdot (b_0, b_1, \ldots, b_{m-1}) = (c_0, c_1, \ldots, c_{m-1})$, where c_k, for $0 \le k \le m-1$, is computed by as follows. First, we have that equality written as: $\sum_{k=0}^{m-1} c_k \beta^{2^k} = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j \beta^{2^i} \cdot \beta^{2^j}$. We will compute the products $\beta^{2^i} \cdot \beta^{2^j} = \sum_{k=0}^{m-1} \lambda_{i,j}^{(k)} \beta^{2^k}$, for all $1 \le i, j \le m-1$, where $\lambda_{i,j}^{(k)} = 0$ or 1. Replacing them back in the previous equality, we have: $c_k = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j \lambda_{i,j}^{(k)}$, where the addition on subscripts are of modulo m. Observe that: $\lambda_{i,j}^{(k)} = \lambda_{i-k,j-k}^{(0)}$. Then we can rewrite (after we dropped the superscript in $\lambda_{i,j}^{(0)}$):

$$c_k = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j \lambda_{i-k,j-k}^{(0)} = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_{i+k} b_{j+k} \lambda_{i,j}^{(0)} = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_{i+k} b_{j+k} \lambda_{i,j}$$

Hence we need only the first term in the expansion of the product

$$\beta^{2^i} \cdot \beta^{2^j} = \lambda_{i,j}\beta + \sum_{k=1}^{m-1} \lambda_{i,j}^{(k)} \beta^{2^k}.$$

This formula for c_k is very helpful in hardware implementation. Indeed, each term c_k can be obtained from the term c_0 by the same hardware setup and a left k-cyclic shift of the involved variables; that is, adding k to each subscript in the formula for c_0. In other words, $(a_0, a_1, \ldots, a_m)$ and $(b_0, b_1, \ldots, b_m)$ are replaced by $(a_k, a_{k+1}, \ldots, a_m, a_0, a_1, \ldots, a_{k-1})$ and $(b_k, b_{k+1}, \ldots, b_m, b_0, b_1, \ldots, b_{k-1})$, respectively.

The complexity m_λ of a normal basis is the number of non-zero terms $\lambda_{i,j}$. Then we have $2m - 1 \le m_\lambda \le m^2$. When $m_\lambda = 2m - 1$, the normal basis is an optimal normal basis that will be discussed next. It is the case that $\lambda_{0,j} = 1$ for precisely one j, $0 \le j \le m - 1$ and that for each i, $0 \le i \le m - 1$, $\lambda_{i,j} = 1$ for precisely two distinct values j, $0 \le j \le m - 1$. This is the most important and popular normal basis used in cryptography.

Inversion: One of the most obvious ways (and also efficient in some setups) is to convert to an inversion on a more familiar polynomial basis representation by a basis-change matrix multiplication and convert the result back to the original normal basis to display.

Deutsch, Omura, Reed, Shao, Truong & Wang [DORSTW85] proposed a new method for inversion, which is derived from the equality: $a^{-1} = a^{2^m-2}$, $\forall a \ne 0$. Then the exponent of a can be manipulated as: $2^m - 2 = 2\sum_{s=0}^{m-2} 2^s = \sum_{s=1}^{m-1} 2^s$. Hence, $a^{-1} = a^{2^m-2} = a^{\sum_{s=1}^{m-1} 2^s} = \prod_{s=1}^{m-1} a^{2^s}$, where we already knew how to compute efficiently each term a^{2^s}. Considering that $(m - 1)$ cyclic shift operations are of no cost, we still need $(m - 2)$ multiplications in the finite field $GF(2^m)$ for the final product. This method is impractical for large m.

The most efficient algorithm was proposed by Itoh & Tsujii [IT88] or Itoh, Teechai & Tsujii [ITT86]. They exploited a factorization of the exponent $(2^m - 2)$ into interweaved cyclic shifts (i.e., squares) and multiplications. We can write: $2^m - 2 = 2\,(2^{m-1} - 1)$, where $2^{m-1} - 1 = \begin{cases} (2^{(m-1)/2} - 1)(2^{(m-1)/2} + 1) \text{ if } m \text{ is odd,} \\ 2(2^{(m-2)/2} - 1)(2^{(m-2)/2} + 1) + 1 \text{ if } m \text{ is even} \end{cases}$. Hence, for m odd, we can compute $\alpha^{2^{m-1}-1}$ from $\alpha^{2^{(m-1)/2}-1}$ with a right $[(m - 1)/2]$-cyclic shift and one multiplication. For m even, we can compute $\alpha^{2^{m-1}-1}$ from $\alpha^{2^{(m-2)/2}-1}$ with a total of $(m/2)$ right 1-cyclic shifts and two multiplications. It can reduce the computation to exactly $\lfloor \log_2(m - 1) + H(m - 1) - 1 \rfloor \le 2\log_2(m - 1)$ multiplications and $(m - 1)$ cyclic shifts, where $H(m)$ is the Hamming weight of the binary representation of m. We can check this formula by induction. However, this method requires storage for intermediate results.

A similar algorithm was proposed independently by Vanstone in a lecture at NTT (Nippon Telegraph and Telephone, Japan) in 1987, and it can also be applied to a general power operation.

Another method for inversion, described in Agnew, Beth, Mullin & Vanstone [ABMV93], does not require storage for intermediate results but requires more multiplications.

If $m - 1 = gh$, then we have $\alpha^{-1} = \alpha^{2(2^{m-1}-1)} = \eta^{2^{gh}-1} = \eta^{(2^g-1)\left(\sum_{i=0}^{h-1} 2^{gi}\right)}$, where $\eta = \alpha^2$ and we used the identity: $a^h - 1 = (a - 1) \sum_{i=0}^{h-1} a^i$, where $a = 2^g$, for the exponent of η. Let denote $\sum_{i=0}^{h-1} 2^{gi} = T$. Then we can compute two factors:

(*i*) $\beta = \eta^{2^g-1} = \eta^{2^0+2^1+\cdots+2^{g-1}} = \prod_{i=0}^{g-1} \eta^{2^i}$ in $(g - 1)$ multiplications (together with $(g - 1)$ right 1-cyclic shifts) and

(*ii*) $\beta^T = \prod_{i=0}^{h-1} \beta^{2^{gi}}$ in $(h - 1)$ multiplications and $(h - 1)g$ right 1-cyclic shifts.

Hence, in total, we can compute the inverse α^{-1} of α in $(g + h - 2)$ multiplications and in $hg - 1 = m - 2$ right 1-cyclic shifts. This number is minimized when both g and h are about $(m - 1)^{1/2}$.

b. Miscellaneous implementations

Trace of an element: Given a point $a = (a_0, a_1, \ldots, a_{m-1})$ in a normal basis representation, its trace will be easily computed as: $Tr(a) = a_0 \oplus a_1 \oplus \ldots \oplus a_{m-1}$. Indeed, from the squaring formula discussed above, we have

$$\begin{aligned} Tr(a) &= a + a^{2^1} + a^{2^2} + \cdots + a^{2^{m-1}} \\ &= (a_0, a_1, \ldots, a_{m-1}) + (a_{m-1}, a_1, \ldots, a_{m-2}) + \ldots + (a_2, a_3, \ldots, a_1) + (a_1, a_2, \ldots, a_0) \\ &= (x, x, \ldots, x), \text{ where } x = a_0 \oplus a_1 \oplus \ldots \oplus a_{m-1}. \end{aligned}$$

We observe: $Tr(a) = 0$ if $x = 0$ and $Tr(a) = 1$ if $x = 1$. Hence $Tr(a) = a_0 \oplus a_1 \oplus \ldots \oplus a_{m-1}$.

Therefore, the trace function is also a parity function that indicates whether the number of bits 1 is odd or even. Furthermore, since squaring and square root operations are just cyclic shifts, we have a trivial proof for the identity: $Tr(a) = Tr(a^2) = Tr(a^{1/2})$.

Finding the roots of equation $x^2 + x + b = 0$: We can observe again that since $Tr(x + x^2) = 0$, the equation has solutions only when $Tr(b) = 0$. In that case, one root is x, then the other root is $(x + 1)$, since $(x + 1)^2 + (x + 1) = x^2 + 1 + x + 1 = x^2 + x$.

In normal basis representation, the roots can be found easily by bit operations. Indeed, we can write $x = (x_0, x_1, \ldots, x_{m-1})$ and $x^2 = (x_{m-1}, x_0, x_1, \ldots, x_{m-2})$. Hence the equation is: $x^2 + x = (x_0 \oplus x_{m-1}, x_1 \oplus x_0, \ldots, x_{m-1} \oplus x_{m-2}) = (b_0, b_1, \ldots, b_{m-1}) = b$. Then one can choose $x_{m-1} = 0$ to compute other bits of a root $x = (x_0, x_1, \ldots, x_{m-1})$. The other root is for $x_{m-1} = 1$. Then we have: $x_0 = b_0 \oplus x_{m-1}$, $x_1 = b_1 \oplus x_0$, $x_2 = b_2 \oplus x_1$, …, $x_{m-2} = b_{m-2} \oplus x_{m-3}$.

More generally, we can solve the equation $x^2 + ax + b = 0$, where a is an invertible element, by transferring it to the form $X^2 + X + a^{-2}b = 0$, where $X = a^{-1}x$.

Discrete exponentiation: The goal is to compute the power $a^n \in GF(2^m)$ where n is its binary form $n = (n_{s-1},\ldots, n_1, n_0) = \sum_{i=0}^{s-1} n_i 2^i$.

In a regular way, we then write: $a^n = \prod_{i=0}^{s-1} a^{n_i 2^i} = \prod_{i=0}^{s-1} (a^{2^i})^{n_i}$. Using normal basis, we get terms a^{2^i} simply by cyclic shifts. Thus we need only a number of $\left(\sum_{i=0}^{s-1} n_i\right) - 1$ multiplications or probabilistically about $(r/2)$ multiplications.

Following the 2^d-ary expansion method, we may pad an extra number of bit 0's to the left side of the bit string to make $s = dr$ for some integer r. Then we have n of the form $n = (N_{r-1},\ldots, N_1, N_0)$, where each N_i is a d-bit string and $N_{r-1} \neq (0\ldots0)$. Explicitly, we can write: $n = \sum_{j=0}^{r-1} N_j 2^{dj}$ and $N_j = \sum_{i=0}^{d-1} n_{jd+i} 2^i$. Let w_j be the value of the d-bit string N_j. The idea is to rewrite n as a sum over w_j, the values of the d-bit strings N_i. We have: $1 \leq w_j \leq 2^d - 1$, for any j. Then $n = \sum_{w=1}^{2^d-1} \left(\sum_{j=0}^{r-1} s_{j,w} 2^{dj} \right) w = \sum_{w=1}^{2^d-1} W(w) w$, where $s_{j,w} = 1$ if $w_j = w$ and $s_{j,w} = 0$ if otherwise. Hence $a^n = \prod_{w=1}^{2^d-1} \left(a^{W(w)}\right)^w$, where $W(w) = \sum_{j=0}^{r-1} s_{j,w} 2^{dj}$.

For a randomly chosen n, the term $W(w)$ will have about $(r/2^d)$ non-zero terms in it. In a normal basis, $a^{W(w)}$ will be computed in $(r/2^d) - 1$ multiplications. Then computing the term $(a^{W(w)})^w$ will need probabilistically about $(d/2) - 1$ multiplications. We need to compute all $(2^d - 1)$ such terms $(a^{W(w)})^w$; and it is possible to use $(2^d - 1)$ microprocessors in parallel. Finally, $(2^d - 2)$ multiplications will provide the final value of a^n.

Refer to Agnew, Beth, Mullin & Vanstone [ABMV93] for further analysis on the numbers of operations. Other works were proposed by Cohen, Miyaji & Ono ([CMO97], [CMO98]).

4.A.4. Optimal normal bases (ONB) (Gao & Lenstra [GL92])

a. Existence of ONBs

An optimal normal basis over F_2 only exists in finite field $GF(2^m)$ for certain values of m. Recall from the previous section, when the complexity m_λ of a normal basis (i.e., the number of non-zero terms $\lambda_{i,j}$) is equal to $(2m - 1)$, then the normal basis is called an optimal normal basis. There are two types of ONB, type I and type II, depending on m and hence on the mathematical formulae defining them.

Theorem (Mullin, Onyszchuk, Vanstone & Wilson [MOVW89])

(*i*) *The finite field* $GF(2^m)$ *has an optimal normal basis if and only if* $(m + 1)$ *is a prime and* 2 *is a primitive element in the finite field* F_{m+1}. (Type I ONB).

(*ii*) *If* $(2m + 1)$ *is a prime and* 2 *is a primitive element in the finite field* F_{2m+1}, *then the finite field* $GF(2^m)$ *has an optimal normal basis.* (Type II ONB).

(*iii*) *If* $(2m + 1)$ *is a prime such that* $2m + 1 \equiv 3 \pmod 4$, *and* 2 *generates the quadratic residues in the finite field* F_{2m+1}, *then the finite field* $GF(2^m)$ *has an optimal normal basis.* (Type II ONB).

The converse of the last two statements is also true. Namely,

If the finite field $GF(2^m)$ has a Type II optimal normal basis, then $(2m + 1)$ is a prime and either

2 is a primitive element in the finite field F_{2m+1}, or

$2m + 1 \equiv 3 \pmod 4$ and 2 generates the quadratic residues in the finite field F_{2m+1}. (It is useful to observe that $2m + 1 \equiv 3 \pmod 4$ if and only if m is odd.)

All ONB's of Type I or II are obtainable by the three statements in the above theorem. The conjecture is that: If m does not satisfy the criteria in the above three statements, then the finite field $GF(2^m)$ does not contain an optimal normal basis. In a later section, we will discuss other low-complexity normal bases, where $m_\lambda > 2m - 1$.

For a more explicit and practical approach to establish algorithms for ONB, we consider further analyses on m required by the above theorem.

Statement (*i*): Let $s = m + 1$. Recall that 2 is not primitive in F_s, for prime s satisfying $s \equiv \pm 1 \pmod 8$. Hence we are interested in case of prime $s \equiv 3$ or $5 \pmod 8$ only when checking the existence of Type I ONB.

When $m \equiv 2$ *or* $4 \pmod 8$ *and* $s = m + 1$ *is prime: Type I ONB exists if and only if* $\text{ord}_s(2) = s - 1 = m$.

This case is proved in a more general finite field $GF(p^m)$ for any prime p. The requirement, that $s = m + 1$ is prime, causes quite a few finite fields having no ONB of type I, such as when m is odd.

The last two statements depend on whether 2 is primitive (hence, not a quadratic residue) in F_p or not primitive.

Statement (*ii*): Let $s = 2m + 1$. We consider the cases that 2 is primitive in F_s, i.e., when $s \equiv 3$ or $5 \pmod 8$, or, equivalently, $m \equiv 1$ or $2 \pmod 4$.

When $m \equiv 1$ *or* $2 \pmod 4$ *and* $s = 2m + 1$ *is prime: Type II ONB exists if and only if we have:* $\text{ord}_s(2) = s - 1 = 2m$.

Observe that if $\text{ord}_s(2) = s - 1$, then $2^{2m} \equiv 1 \bmod(2m + 1)$, or $(2m + 1) \mid (2^{2m} - 1)$. The finite field $GF(2^{2m})$ contains a primitive $(2m + 1)^{\text{th}}$ root of unity, called β. It generates an optimal normal basis of $GF(2^{2m})$ over F_2. Moreover, let $\varepsilon = \beta + \beta^{-1} \in GF(2^m)$. Then ε will generate an optimal normal basis of $GF(2^m)$ over F_2.

Statement (*iii*): Let $s = 2m + 1$. We consider the cases that 2 is a quadratic residue in the finite field F_s, i.e., when $s \equiv \pm 1 \pmod 8$. We need also $s \equiv 3 \pmod 4$. Hence $m \equiv 3 \pmod 4$ is the only case to be considered.

When $m \equiv 3 \pmod 4$ *and* $s = 2m + 1$ *is prime: Type II ONB exists if and only if* $\text{ord}_s(2) = (s - 1)/2 = m$.

Observe that if $ord_s(2) = (s - 1)/2$, then $2^m \equiv 1 \bmod(2m + 1)$, or $(2m + 1) \mid (2^m - 1)$. The finite field $GF(2^m)$ contains a primitive $(2m + 1)^{\text{th}}$ root of unity, called β. It generates an optimal normal basis of $GF(2^m)$ over F_2.

b. Structure of ONBs (Ash, Blake & Vanstone [ABV89])

Type I ONB uses the reduction polynomial (or field polynomial) of the form $f(x) = x^m + x^{m-1} + \ldots + x + 1$. Let β be a primitive $(m + 1)^{\text{th}}$ root of unity in $GF(2^m)$. Then β generates the Type I ONB. Since $\beta^{m+1} - 1 = 0$ and $\beta \neq 1$, therefore $\beta^m + \beta^{m-1} + \ldots + \beta + 1 = f(\beta) = 0$. Hence β is a root of the reduction polynomial. We also have $f(\beta^{2^i}) = \left(f(\beta)\right)^{2^i} = 0$. Hence all elements in the normal basis are exactly all roots of $f(x)$, that is now also called normal polynomial.

Property: *There exists an integer k_i such that* $1 \le k_i \le m-1$, *such that* $\beta \cdot \beta^{2^i} = \beta^{2^{k_i}}$, *with the exception of one value of i such that* $1 \le i \le m-1$.

Type II ONB uses the reduction (normal) polynomial $f(x) = f_m(x)$, where all polynomials $f_i(x)$ of degree i are computed by the recursive formulae (modulo 2) for all $0 \le i \le m$: $f_0(x) = 1$, $f_1(x) = x + 1$ and $f_{i+1}(x) = x f_i(x) + f_{i-1}(x)$, (mod 2), for $i \ge 1$.

Let β be a primitive $(2m+1)^{\text{th}}$ root of unity in $GF(2^{2m})$. Then $\varepsilon = \beta + \beta^{-1}$ generates the ONB for $GF(2^m)$. This idea is also used to create other low complexity normal bases. When $m \equiv 3 \pmod 4$, β is in fact in $GF(2^m)$. Thus $\varepsilon = \beta$ can generate the ONB for $GF(2^m)$.

Property: *There are integers k_i and m_i such that* $1 \le k_i, m_i \le m-1$, *such that* $\varepsilon \cdot \varepsilon^{2^i} = \varepsilon^{2^{k_i}} + \varepsilon^{2^{m_i}}$, *for any i such that* $1 \le i \le m-1$.

Note that the generator for the ONB of a finite field is not necessarily a generator for non-zero elements of the finite field itself.

Uniqueness: For every binary finite field $GF(2^m)$, there is at most one ONB of each type.

Normal bases for a finite field $GF(p^m)$ where p is an odd prime

A normal basis of $GF(p^m)$ is a basis of the form $\{\beta, \beta^p, \beta^{p^2}, \ldots, \beta^{p^{m-1}}\}$, where $\beta \in GF(p^m)$. Such a basis always exists. Then $a = (a_0, a_1, \ldots, a_{m-1})$ will represent the element $a = a_0\beta + a_1\beta^p + a_2\beta^{p^2} + \cdots + a_{m-1}\beta^{p^{m-1}}$. Particularly, we can write the zero element $0 = (0, 0, \ldots, 0)$ and multiplicative identity element $1 = (1,1,\ldots,1)$.

Given a point $a = (a_0, a_1, \ldots, a_{m-1})$ represented in a normal basis, we have:

$$a^p = \left(\sum_{i=0}^{m-1} a_i \beta^{p^i} \right)^p = \sum_{i=0}^{m-1} a_i^p \beta^{p^{i+1}} = \sum_{i=0}^{m-1} a_{i-1} \beta^{p^i} = (a_{m-1}, a_0, a_1, \ldots, a_{m-2})$$

Theorem (Mullin, Onyszchuk, Vanstone & Wilson [MOVW89])

Suppose that the finite field $GF(p^m)$ contains $(m+1)^{\text{th}}$ roots of unity. If the m non-unit roots of unity are linearly independent, then $GF(p^m)$ contains an optimal normal basis.

One can write $\{\beta, \beta^p, \beta^{p^2}, \ldots, \beta^{p^{m-1}}\}$, where β is a primitive $(m+1)^{\text{th}}$ root of unity. The elements of the set are called conjugates of β. In other words, we now state:

The finite field $GF(p^m)$ contains an optimal normal basis consisting of m non-unit $(m+1)^{\text{th}}$ roots of unity if and only if $(m+1)$ is a prime and p is primitive in the finite field F_{m+1}.

Here is a sketch of the proof. If $(m+1)$ is prime then $p^{(m+1)-1} \equiv 1 \bmod (m+1)$. That is, $(m+1) \mid (p^m - 1)$, where $(p^m - 1)$ is the order of the multiplicative subgroup $GF(p^m)^*$. Then the finite field $GF(p^m)$ contains a primitive $(m+1)^{\text{th}}$ root β of unity. Since p is primitive in Z_{m+1}, the minimal polynomial of β is $(x^{m+1} - 1)/(x - 1)$ and the non-unit $(m+1)^{\text{th}}$ roots are linearly independent. Hence the finite field $GF(p^m)$ contains an optimal normal basis generated by element β. If finite field $GF(p^m)$ contains an optimal normal basis consisting of m non-unit $(m+1)^{\text{th}}$ roots of unity, then $(m+1)$ must be prime. Hence we have: $p^{(m+1)-1} \equiv 1 \bmod (m+1)$, or $p^m \equiv 1 \bmod (m+1)$. ⁂

⁂ **Number theory tip – Primitive elements of a finite field**

4.A.5. Low-complexity normal bases for $GF(2^m)$ (or Gaussian normal bases) (Ash, Blake & Vanstone [ABV89])

Recall that the complexity m_λ of a normal basis is the number of non-zero terms $\lambda_{i,j}$. When the optimal normal basis does not exist, we could use the low-complexity normal basis (or Gaussian normal basis,) where $m_\lambda > 2m - 1$.

The ideas of Ash, Blake & Vanstone [ABV89] are from the construction of Type II ONB previously described. For a finite field $GF(2^m)$, we find a small integer k such that $(km + 1)$ is a prime. Under certain conditions, there will exist some element $\beta \in GF(2^{km})$, $\beta \neq 1$ and $\beta^{km+1} = 1$. We will try to apply on β a trace-like operator from the finite field $GF(2^{km})$ to $GF(2^m)$ in order to find a generator for a low complexity normal basis of $GF(2^m)$. (In case of Type II ONB, we have $k = 2$ and the operator is $\varepsilon = \beta + \beta^{-1}$.)

When m is not divisible by 8, one can use the Gaussian cyclotomic periods to construct easily an efficient normal basis for $GF(2^m)$. We summarize the results into this theorem.

Theorem: *Given a type T, that is a positive integer, suppose m is not divisible by* 8*, and* $p = Tm + 1$ *is a prime. Let* $k = \text{ord}_p(2)$, *and* $h = Tm/k = (p - 1)/k$. *The finite field* $GF(2^m)$ *has a normal basis of type T if and only if* $\gcd(h, m) = 1$.

When $T = 1$ or 2, it is type I ONB or type II ONB, respectively, which were discussed previously. They are the most popular normal bases that are the most efficient multiplications in finite fields.

When $T \geq 3$, it is called a low-complexity normal base of type T (or Type T Gaussian normal basis.) If both type I and II ONBs do not exist, then the type T Gaussian normal basis of the smallest T should be used.

Uniqueness: For any positive integer T, every binary finite field $GF(2^m)$ has at most one Type T Gaussian normal basis.

Recursive formula for reduction polynomial

Let u be an integer of order T modulo p: $\text{ord}_p(u) = T$. For $1 \leq i \leq m$, let us define

$$Z_i(z) = z^{2^{i-1} \pmod p} + z^{2^{i-1} \cdot u \pmod p} + z^{2^{i-1} \cdot u^2 \pmod p} + \cdots + z^{2^{i-1} \cdot u^{T-1} \pmod p}.$$

Let $f_0(t,z) = 1$, $Z(z) = z^{p-1} + z^{p-2} + \ldots + z + 1$, and $f_i(t, z) = (t + Z_i(z))\, f_{i-1}(t, z) \bmod Z(z)$. Then the reduction polynomial for the normal basis is: $f(t) = f_m(t, z)$.

A primitive element of a finite field F_q is an element α such that its order is $(q - 1)$ and every non-zero element of F_q is a power of α. Thus, a primitive element is a generator of the multiplicative subgroup $GF(q)^*$.

Let q be an integer ≥ 2. If there is some integer $a < q$, such that the order of a modulo q is $(q - 1)$, then q is a prime number. Thus, q is a prime if and only if there is an element of order exactly $(q - 1)$ (modulo q).

There are exactly $\phi(p - 1)$ primitive elements in the prime finite field F_p, where ϕ denotes the Euler's ϕ-function.

Primitive elements & quadratic residues

In general, if x is a quadratic residue, then we can write $x = y^2$, for some $y \in F_p$. Hence $x^{(p-1)/2} = y^{p-1} = 1$. Therefore, x cannot be primitive. In other words,

If x is primitive, then x is not a quadratic residue. In particular, the quadratic residues in F_p *are of the form* x^{2n} *and the quadratic non-residues are of the form* x^{2n+1}.

Note that we can write $(z - 1)Z(z) = z^p - 1$. Hence the modulo $Z(z)$ can be performed simply by: $z^p \equiv 1 \bmod Z(z)$, $z^{p+1} \equiv z \bmod Z(z)$, and generally, $z^{p+j} \equiv z^j \bmod Z(z)$.

More calculations for other small finite fields $GF(2^m)$ are given in the table 4.2. as illustrated examples.

m	Computing polynomials for ONB Type II of finite field $GF(2^m)$	Existence of Type I & II ONB
2	$f_2(t) = t^2 + t + 1$ (The only irreducible polynomial of degree 2 over F_2)	Type I ≡ Type II
3	$f_3(t) = t(t^2 + t + 1) + t + 1 = t^3 + t^2 + 1$	Type II, no Type I
4	$f_4(t) = t(t^3 + t^2 + 1) + t^2 + t + 1 = t^4 + t^3 + t^2 + 1$ (reducible)	Type I, no Type II
5	$f_5(t) = t(t^4 + t^3 + t^2 + 1) + t^3 + t^2 + 1 = t^5 + t^4 + t^2 + t + 1$	Type II, no Type I
6	$f_6(t) = t(t^5 + t^4 + t^2 + t + 1) + t^4 + t^3 + t^2 + 1 = t^6 + t^5 + t^4 + t + 1$	Type II, no Type I
7	$f_7(t) = t(t^6 + t^5 + t^4 + t + 1) + t^5 + t^4 + t^2 + t + 1 = t^7 + t^6 + t^4 + 1$ (reducible)	no Type I no Type II
8	$f_8(t) = t(t^7 + t^6 + t^4 + 1) + t^6 + t^5 + t^4 + t + 1 = t^8 + t^7 + t^6 + t^4 + 1$	no Type I no Type II
9	$f_9(t) = t^9 + t^8 + t^6 + t^5 + t^4 + t + 1$	Type II, no Type I
10	$f_{10}(t) = t^{10} + t^9 + t^8 + t^5 + t^4 + t^2 + t + 1$ (reducible)	Type I, no Type II
11	$f_{11}(t) = t^{11} + t^{10} + t^8 + t^4 + t^3 + t^2 + 1$	Type II, no Type I

Table 4.2. Computing reduction polynomials for some ONB Type II's

4.A.6. Self-dual bases and self-dual normal bases

a. Dual bases

Two bases $B = \{b_0, b_1, \ldots, b_{m-1}\}$ and $C = \{c_0, c_1, \ldots, c_{m-1}\}$ of a finite field $GF(p^m)$ over F_p are called dual bases (or complementary bases) if and only if

$$Tr(c_i.b_j) = \delta_{ij}, \text{ where } \delta_{ij} = 0 \text{ if } i \neq j \text{ and } \delta_{ij} = 1 \text{ if } i = j.$$

The symbol δ_{ij} is called the Kronecker delta function.

Fact: *Every basis has a unique dual basis*.

Let $s = s_0b_0 + s_1b_1 + \ldots + s_{m-1}b_{m-1}$ and $t = t_0c_0 + t_1c_1 + \ldots + t_{m-1}c_{m-1}$. Then the trace of the product $(s.t)$ is defined by: $Tr(st) = s_0t_0 + s_1t_1 + \ldots + s_{m-1}t_{m-1}$, and is used to define a non-degenerate symmetric bilinear form: $\langle s, t \rangle = Tr(st)$, which is called the trace bilinear form.

Theorem: *Every element $a \in GF(q^m)$ can be expressed in the dual basis C as:* $a = \sum_{i=0}^{m-1} Tr(ab_i)c_i$.

The above fact still true in general, when the trace function $Tr(\cdot)$ is replaced by any nontrivial linear function f (or transformation) from the finite field $GF(p^m)$ to the finite field F_p. Dual bases have many advantages in hardware designs but they are more complex in software implementations, such as field squaring. Current works using specially designed dual bases were discussed in literature such as circular dual bases and optimal dual bases.

b. Optimal dual bases

Benaissa, Fenn & Taylor ([BFT96],[BFT96a]) proposed an optimal dual basis of a given polynomial basis with respect to a linear function g. The authors showed that any linear function g can be of the form

$g(z) = Tr(bz)$, $\forall z \in GF(2^m)$, for some element $b \in GF(2^m)$.

Hence they can choose an optimal element b such that the elements of the resulting dual basis are just a permutation of elements in the given polynomial basis.

c. Circular dual bases

Lee & Lim [LL98] proposed the circular dual basis that is the dual basis of a polynomial basis whose reduction polynomial is of the form $f(x) = x^m + \ldots + x + 1$, when it is an irreducible polynomial, for proper values of m. This polynomial basis is the Type I ONB. In fact, this circular basis exist only in the finite fields, which have Type I ONB. Explicitly, if the polynomial basis (or Type I ONB) is $\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\}$, then its circular dual basis is $\{1, c, c^2, \ldots, c^{m-1}\}$, where $c^i = \alpha + \alpha^{-i}$, for $1 \le i \le m-1$. A short list of degree m of such finite fields are: m = 4, 10, 12, 18, 28, 36, 52, 58, 66, 82, 100, 106, 130, 138, 148, 162, 172, 178, 180, 196, 210, 226, 268, 292, 316, 346, 348…

The authors showed a few advantages and efficiencies of this circular dual basis, which should inherit benefits of both dual basis and optimal normal basis, in finite field arithmetic implementations: multiplications, squarings, inversions and basis changes.

d. Self-dual (or self-complementary) bases

A basis $\{e_0, e_1, \ldots, e_{m-1}\}$ is called self-dual (or self-complementary) basis with respect to the trace function $Tr(\cdot)$, if and only if $Tr(e_i e_j) = \delta_{ij}$.

Self-duality will always mean "with respect to the trace function $Tr(\cdot)$," unless stated otherwise.

Property: *Any polynomial basis cannot be a self-dual basis.*

As observed above, a polynomial basis can be a dual basis of itself (or of a permutation of it) with respect to some linear function other than the trace function $Tr(\cdot)$.

Theorem ([LS80])

A finite field $GF(q^m)$ has a self-dual basis over F_q if and only if either q is even, or both q and m are odd.

A basis satisfying only the condition: $Tr(e_i e_j) \neq 0$ if and only if $i = j$, is called a trace-orthogonal basis. Then any finite field $GF(q^m)$ always has a trace-orthogonal basis over the field F_q.

e. Self-dual normal bases

A normal basis that is also self-dual is called self-dual normal basis. A finite field $GF(2^m)$ can have more than one self-dual normal basis. For instance, a Type II ONB is a self-dual normal basis.

Theorem (MacWilliams & Sloane [MS77])

If m is odd, then the finite field $GF(2^m)$ has a self-dual normal basis over F_2.

Theorem (Existence theorem): *A finite field $GF(q^m)$ has a self-dual normal basis over F_q if and only if either q is even and $m \not\equiv 0 \pmod 4$, or both m and q are odd.*

The necessary condition is trivial for q odd, and due to Imamura & Morii [IM85]. The sufficient condition is due to Lempel & Weinberger [LW88].

Jungnickel, Menezes & Vanstone [JMV90] counted the number of self-dual bases and for case q prime only, the number of self-dual normal bases. Later, Beth & Geiselmann extended the later result to any finite field. Geiselmann & Gollmann [GG90] also mentioned that dual basis multipliers have some advantages when circuits are designed from certain standard cells (e.g., TTL), but there seems to be no reason to prefer

them in full custom VLSI-design. Deutsch, Hsu, Reed & Truong [DHRT88] gave a comparison on VLSI multipliers using differents bases.

4.A.7. Primitive normal bases

A normal basis $\{\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{m-1}}\}$ of the finite field $GF(q^m)$ over a finite field F_q where q is any prime power, is called a primitive normal basis if β is a primitive root of the multiplicative subgroup $GF(q^m)^*$.

Theorem (Lenstra & Schoof [LS87])

For every prime power $q > 1$, and every positive integer m, there exists a primitive normal basis of finite field $GF(q^m)$ over F_q.

Carlitz ([C52],[C52a]) proved this result with the case q prime and q^m sufficiently large. Davenport [D69] extended to all m with $q = p$ prime. The author also showed that the number of choices for β is at least $(p-1)^m$.

4.A.8. Non-conventional basis

a. All-one-polynomial (AOP) basis (Hasan & Wu [HW98])

The reduction polynomial $f(x) = x^m + x^{m-1} + \ldots + x + 1$ whose coefficients are all 1 is called all-one-polynomial (AOP). Let $\alpha = x + (f(x))$ be a root of $f(x)$. Hence the set $\{\alpha, \alpha^2, \ldots, \alpha^m\}$ is a non-conventional basis, where $\alpha^i = x^i + (f(x))$, for $1 \le i \le m$.

b. Equally Spaced Polynomial (ESP) basis (Itoh [I91] and Hasan & Wu [HW98])

A polynomial $g(x) = x^{sm} + x^{s(m-1)} + \ldots + x^s + 1 = f(x^s)$ over $GF(2)$, where $f(x)$ is a AOP of degree m over $GF(2)$ is called an s-equally spaced polynomial (s-ESP) of degree sm.

4.A.9. The choice of bases

The security of an elliptic curve cryptosystem does not depend on the choice of basis representation for the finite field $GF(2^m)$. Polynomial bases and optimal normal bases are equally secure. Moreover, all the bases are mutually transformable by using basis-change matrix multiplication, we can also use flexibly one basis for internal calculations and another basis for outputting data.

Any standard book on linear algebra always discusses the problem of using matrix multiplication for basis change or conversion. The implementation should be rather simple. The storage requirement is known to be about $O(m^2)$ bits over a binary finite field $GF(2^m)$.

Kaliski & Yin ([KY98],[KY98p]) and Kaliski & Liskov [KL98] discussed new algorithms for basis conversion over finite field $GF(2^m)$. These algorithms require only $O(m)$-bit storage.

For software, the polynomial bases could be easier to understand and more efficient. But the normal bases are more efficient for hardware implementation by taking advantage of the fact that squaring operation is simply a cyclic shift, while in polynomial bases, implementation of the squaring operation cannot be easier than that of multiplication.

However, this advantage must be exchanged for a larger and more complicated layouts for multiplications, unless one uses the optimal normal bases. We also take into account that the easy squaring operations also reduce the number of multiplications in scalar point multiplication, [GG90].

4.A.10. Comparisons of finite fields

In implementing elliptic curve cryptosystems on finite fields, the binary finite fields $GF(2^m)$ are preferred over the prime finite fields F_p, even though F_p has performance advantages in software since it is rather more comprehensive.

The arithmetic in the binary finite field $GF(2^m)$ is easier to implement than it is in the prime finite field F_p, especially when optimal normal bases are used. The easy squaring operations also reduce the number of multiplications in scalar point multiplication.

Point compression & recovery or compact point techniques help in reducing the encrypted message.

In some cryptosystems where the supersingular elliptic curves could be used, we can also reduce the number of arithmetic operations.

The ability to select the underlying finite field and its basis to optimize the finite field operations is also an advantage of an elliptic curve cryptosystem over other systems based on the discrete logarithm problem or the integer factoring problem.

4.A.11. Composite extension finite fields and subfields

When m is a composite number, $m = rs$, then the composite extension finite field $GF(2^m)$ can be considered an extension field of degree s over finite field $GF(2^r)$. The finite field $GF(2^r)$ is called a subfield of $GF(2^m)$.

The elliptic curve over a subfield is also used in computing the order of an elliptic curve over a composite extension finite field using Hasse-Weil's theorem.

We can also represent elements in a composite extension finite field over its subfield using either one of the two bases discussed earlier.

We should point out that the finite field $GF((2^r)^s)$ is isomorphic to the finite field $GF(2^m)$, but their field operations (additions and multiplications) are different depending on the irreducible field polynomials, $P(x)$ of $GF((2^r)^s)$ over $GF(2^r)$ and $Q(y)$ of $GF(2^r)$ over $GF(2)$.

It also depends on the posbbile factorizations of m (other than factors r and s). The choice of those field polynomials are essential to determine the algorithmic complexity of arithmetic operation of $GF((2^r)^s)$.

Using polynomial bases: Let $\{r_{s-1},\ldots, r_1, r_0\}$ be a polynomial basis for $GF(2^m)$ over $GF(2^r)$. Every element a in the finite field $GF(2^m)$ can be uniquely written in the form $a = (c_{s-1},\ldots, c_1, c_0) = \sum_{i=0}^{s-1} c_i r_i$, or by the polynomial of the form $a = c_{s-1}x^{s-1} + \ldots + c_1 x + c_0$, where each term $c_i \in GF(2^r)$, for $s - 1 \geq i \geq 0$, is also represented in a polynomial basis over F_2.

Using normal bases: Let $\{\beta, \beta^{2^r}, \beta^{2^{2r}},\ldots, \beta^{2^{2(s-1)r}}\}$ be a normal basis for $GF(2^m)$ over $GF(2^r)$. Then every element a in the binary finite field $GF(2^m)$ will be represented by $a = (c_0, c_1,\ldots, c_{s-1}) = c_0\beta + c_1 \beta^{2^r} + c_2 \beta^{2^{2r}} + \ldots + c_{s-1}$, where $c_i \in GF(2^r)$, for $0 \leq i \leq s - 1$. In this representation, we observe that $c_i^{2^r} = c_i$, then

$$a^{2^r} = c_{s-1}\beta + c_0 \beta^{2^r} + c_1 \beta^{2^{2r}} + c_2 \beta^{2^{3r}} + \ldots + c_{s-2} \beta^{2^{(s-1)r}} .$$

We do not have the rule "squaring is a right 1-cyclic shift" anymore. Instead, the 2^r-th power of an element is a right cyclic shift of coefficients c_i. This implementation is most useful when, of course, $r = 1$ only.

Finally, the elements $c_i \in GF(2^r)$ also can be presented by either basis over F_2. Recall that the choice of bases is not a security factor. One can always converse simply from one basis to another.

The arithmetic operations in the composite extension finite field $GF(2^m) = GF((2^r)^s)$ over the finite field $GF(2^r)$ do not only involve modulo 2 as in the case over F_2 but also involve arithmetic operations in the finite field $GF(2^r)$ over F_2. There are many research efforts in techniques to maximize performing operations over subfields.

The following lemma focused on optimal normal bases rather than on normal bases. Refer to Mullin [M93] or Agnew, Mullin & Vanstone [AMV93].

Lemma: *There is an optimal normal basis for finite field* $GF(2^{rs})$ *over* $GF(2^r)$ *if and only if* $gcd(r, s) = 1$ *and there is an optimal normal basis for the finite field* $GF(2^s)$ *over* F_2.

In fact, let $B = \{\alpha, \alpha^2, \alpha^{2^2}, \ldots, \alpha^{2^{(s-1)}}\}$ be an optimal normal basis for the finite field $GF(2^s)$ over F_2. Since gcd $(r, s) = 1$, then B is also linearly independent over $GF(2^r)$ and is an optimal normal basis of the finite field $GF(2^{rs})$ over $GF(2^r)$. The set of elements

$$a = d_0\alpha + d_1\alpha^2 + d_2\alpha^{2^2} + \ldots + d_{s-1}\alpha^{2^{(s-1)}}, \text{ where } d_i \in GF(2^r).$$

has cardinality 2^{rs}; hence it is just the finite field $GF(2^{rs})$.

Refer to Green & Taylor [GT74], Guajardo [G97], Guajardo & Paar [GP97] and Smart [S01] for more discussions on implementations and security issues.

4.A.12. Optimal extension fields (OEF) (Bailey & Paar, [BP98],[BP99],[BP00])

a. The fields [ℵ]

Finite fields of the form $GF(p^m)$, where $m > 1$ (and typically small $m = 3,\ldots,8$) and p is a big prime of the form $p = 2^n \pm c$, for some small c such that $\log_2 c \le \lfloor n/2 \rfloor$ and an irreducible binomial $P(x) = x^m - w$ exists over $GF(p)$, for some element $a \in GF(p)$.

Such number p is called pseudo-Mersenne prime number. In practice, the prime p is chosen to be a little bit smaller than the word size of the processor. In such finite fields, we can perform efficient subfield multiplication, by reducing a $2n$-bit number to roughly $1.5n$ bit value by "folding" the upper half into the lower half. The small value of c helps to improve the subfield modular reduction. There are two special types of OEF that can provide additional advantages on arithmetic operations.

Type I OEF: $p = 2^n \pm 1$. This field allows for subfield modular reduction with very low complexity. For example, good choices for p are: $2^{31} - 1$ and $2^{61} - 1$. Example of implementation of Type I OEF: $GF((2^{61} - 1)^3)$ whose reduction binomial is $x^3 - 37$.

[ℵ] **Number theory tip**

Theorem: *For an integer* $m \ge 2$, *for* $w \in GF(p)$, *the binomial* $x^m - w$ *is irreducible in* $GF(p)$ *if and only if the following two conditions are satisfied:*

(*i*) *Each prime factor of m divides the order e of element w in GF(p), but does not divide* $(p - 1)/e$;

(*ii*) $p \equiv 1 \pmod 4$ *if* $m \equiv 0 \pmod 4$.

Corollary: *If w is a primitive element for* $GF(p)$ *and m is a divisor of* $(p - 1)$, *then the binomial* $x^m - a$ *is irreducible over* $GF(p)$.

An important trivial result is for the case $m = 2$. The binomial $x^2 - w$ is irreducible over $GF(p)$ when w is a primitive element.

Type II OEF: has an irreducible binomial $x^m - 2$. This field allows for a reduction in the complexity of extension field modular reduction since multiplication by w can be implemented using shiftings. Example of Type II OEF: $GF((2^{63} - 259)^2)$, $GF((2^{63} - 259)^3)$ and $GF((2^{63} - 259)^4)$.

b. The elliptic curves

Over OEFs $GF(p^m)$, the elliptic curves have the form: $y^2 = x^3 + ax + b$, unless for the case $p = 3$, one uses a non-supersingular elliptic curve of the form: $y^2 = x^3 + ax^2 + b$. Refer also to Baier [B01].

c. Implementation issues

The inverse algorithm is the most important algorithm to discuss in implementation works. Over OEFs, authors developed and modified some efficient advanced algorithms implemented in typical finite fields discussed earlier. Refer to Bailey, Paar & Woodbury [BPW00], working over finite field $GF((2^8 - 17)^{17})$ on low-end 8-bit processors and Aoki, Hoshino, Kobayashi, Kobayashi & Morita's [AHKKM]: working on finite field $GF(p^m)$, where prime $p > 3$.

Kim, E. J. Lee & P. J. Lee [KLL98] implemented OEFs using a choice of p less than 2^{16}, which allows for the use of look-up tables for subfield inversion. They also proposed a new inversion algorithm, called MAIA (Modified Almost Inverse Algorithm), which is suited especially for OEFs.

Hoshino, K. Kobayashi, T. Kobayashi & Morita [HKKM99] presented an inversion algorithm for OEFs that is based on a direct solution of a set of linear equations. This method is efficient for small values of m.

4.B. Implementations of elliptic curves

Now, we will discuss issues on implementing elliptic curves. Related problems are choosing an appropriate order, computing the order of a given elliptic curve, and constructing a cryptographically good elliptic curve for a cryptosystem.

4.B.1. Conditions for selecting appropriate elliptic curves

The order of the elliptic curve, $N = \#E(F_q)$, must be divisible by a prime number n that is sufficiently large, $n > 2^{160}$. This is to resist against the Pollard ρ-algorithm.

The order n of a base point P must satisfy the MOV condition: $n \nmid (q^k - 1)$ for all values $k < (\log q)^2$. In practice, $k = 20$ is sufficient.

The best-known attacks on an elliptic curve cryptosystem satisfying these two conditions are a combination of either Pollard-ρ or Shanks' Baby-step-Giant-step and Pollard-Hellman algorithms.

The order of the elliptic curve E must also satisfy the anomalous (or SSA) condition over prime finite fields, $\#E(F_p) \neq p$ to resist the SSA attack.

The choice of point P of order n is not a security factor. In fact, given an elliptic curve, there are many different points having that order which can be chosen.

Cofactor

We denote by n the order of the base point P on the elliptic curve whose order is denoted by N. Usually, we should have either $N = n$ or $N = nl$, where l is a small integer, called a cofactor. In group theory, we call l the index of a subgroup generated by group element P, denoted by $\langle P \rangle$, in the group E.

Recall the inequalities of the order of an elliptic curve, we have: $q + 1 - 2q^{1/2} \le \#E(F_q) = N \le q + 1 + 2q^{1/2}$. Then $(q^{1/2} - 1)^2/n \le N/n = l = (q^{1/2} + 1)^2/n$.

We can observe that l must be an integer and the difference of the upper and lower bounds of l is $d = 4q^{1/2}/n$. Hence if $n > 4q^{1/2}$, then $d < 1$, we must have: $l = \lfloor (q^{1/2} + 1)^2/n \rfloor$.

Therefore, the order of the elliptic curve can be computed as: $\#E(F_q) = N = n.l = n\lfloor (q^{1/2} + 1)^2/n \rfloor$, if there exists a point on E of order $n > 4q^{1/2}$.

Furthermore, the condition $n > 4q^{1/2}$ also implies that $n^2 > (q^{1/2} + 1)^2 \ge N = \#E$. Or $n^2 \nmid N$. Therefore, there is exactly one subgroup of order n in E. This is directly from the well-known Sylow's third theorem in group theory and the subgroup is also called the Sylow n-subgroup.

4.B.2. Methods of constructing elliptic curves

There are four methods for constructing an elliptic curve in cryptography.

Generating random elliptic curves.

Using the Hasse-Weil theorem (on composite extension finite fields).

Using selected orders of elliptic curves or also referred as Complex Multiplication (CM) methods.

Using special elliptic curves such as Koblitz curves.

a. Using a random elliptic curve

We select an elliptic curve at random and compute its order by some algorithm. For special elliptic curves, or over relatively small finite fields, there are a few effective formulas or algorithms for the order. The best-known method for a general case is Schoof's algorithm, together with its improvements and/or extensions.

The advantages:

One can change the elliptic curve as frequently as possible for security reasons.

In order to break into an elliptic curve cryptosystem, the attacker should use an algorithm to solve the ECDLP that can work on any elliptic curve rather than some particular classes of weak elliptic curves.

The disadvantages:

It is tedious and still complicated to use Schoof's algorithm (and even its improved versions) to find an elliptic curve of particular order.

It is more difficult, generally, to implement a random elliptic curve efficiently, while we can optimize implementation on specific elliptic curves, such as in Koblitz curves and some cryptosystems using supersingular elliptic curves.

It is time-consuming to generate an elliptic curve and to perform operations on a general elliptic curve. Other implementations, such as compressing and recovering a point, may need more computations than in some particular cryptosystems.

b. Using Hasse-Weil's theorem

This method is to construct an elliptic curve over a finite field $GF(2^m)$ where m is a composite number. We first construct an elliptic curve over a finite field $GF(2^n)$ for some small factor n of m such that we can compute its order easily. Then we lift it to an elliptic curve over a finite field $GF(2^m)$ where its order can be computed rather easily using the Hasse-Weil's theorem.

We should compute rather easily that $\#E(GF(2^n)) = 2^n + 1 - t$, then we will have $\#E(GF(2^m)) = 2^m + 1 - \alpha^{m/n} - \beta^{m/n}$, where α and β are complex numbers satisfying the

equation $qT^2 - tT + 1 = (1 - \alpha T)(1 - \beta T)$. More practically, $\alpha + \beta = t$ and $\alpha\beta = q = 2^n$. Then the power sum $L_k = \alpha^k + \beta^k$ is the k-th term in the sequence of symmetric functions:

$$L_1 = t,\ L_2 = t^2 - 2q, \text{ and for all } k \geq 3,\ L_k = tL_{k-1} - q\ tL_{k-2}.$$

The sequence L_k is usually called Lucas sequence.

The order $\#E(GF(2^m))$ then has a small factor, namely $\#E(GF(2^n))$. This method of choosing an elliptic curve works on composite extension finite fields and can create only a limited number of elliptic curve orders.

This construction method can increase the performance in generating elliptic curves and doing elliptic curve operations. But in the security aspect, these elliptic curves (also referred as subfield elliptic curves) are considered weak curves.

c. Complex Multiplication (CM) methods

A Complex Multiplication method allows choosing an appropriate elliptic curve order, before constructing explicitly an elliptic curve of that order. In practice, this method is fast, and the big advantage is to eliminate the need for a point counting algorithm. Two methods mentioned in literature are due to Atkin & Morain and Lay & Zimmer.

Atkin-Morain method [Mo91]

The method works on a prime finite field F_p. Recall that $\#E(F_p) = p + 1 - t$, where $t^2 \leq 4p$. It is based on a theorem of the primality-testing algorithm using elliptic curves.

Theorem: *Let p be a prime that can be written as* $4p = t^2 + Ds^2$ *for a given D. Then there exists an elliptic curve E defined over* F_p *such that* $4.\#E(F_p) = (t-2)^2 + Ds^2$.

We call D a Complex Multiplication discriminant for p, or the elliptic curve E has CM property by D, or in fact, by $(-D)^{1/2}$. If we know D for a given curve E, we then can solve for t (and s) in the equation: $4p = t^2 + Ds^2$, and know the order $\#E(F_p)$.

Atkin-Morain algorithm:

Compute $t = p + 1 - \#E(F_p) = p + 1 - N$. Find an integer s and a square-free positive integer D such that $Ds^2 = t^2 - 4p$.

This step can be done since if N is an order of an elliptic curve $\#E(F_p)$, we must have $t^2 \leq 4p$. Hence $A = 4p - t^2 \geq 0$, and it can be written uniquely as $A = Ds^2$, where D is a square-free positive integer. Then we write: $4p = t^2 + Ds^2$.

Construct the Hilbert polynomial $H_D(X)$ of $j(D^{1/2})$, using the above formula.

Find a root r of the equation $H_D(X) \equiv 0 \pmod{p}$.

Create a non-supersingular elliptic curve whose j-invariant is r.

This algorithm can be generalized over finite field F_q, where $q = p^m$.

Although it is possible to choose the order of an elliptic curve before choosing the underlying finite field, in usual cryptographic practice, one prefers to choose the finite field in advance so one can exploit some efficient implementations.

This method usually generates small d. The subset of elliptic curves generated by this method is considerably smaller than the number of elliptic curves available.

Those elliptic curves are thought to be insecure, but no weakness is known so far.

Refer to Miyaji ([Mi91], [Mi93]) for construction examples.

Lay-Zimmer method [LZ94]

This method works on finite fields of characteristic 2 and also over prime finite fields. It solves the following problems:

Given an integer $N > 3$, find a prime p and an elliptic curve $E(F_p)$ such that its order $\#E(F_p) = N$.

Given an integer $k > 1$, decide the existence of a prime $p > 3$ and an elliptic curve $E(F_p)$ such that $E(F_p) \cong Z_k \times Z_k$.

Given two integers m and C, find an elliptic curve E over a finite field $GF(2^m)$, such that its order $\#E = cn$, where n is a prime number and $c \le C$.

Given a prime $p > 3$ and an integer m with $|p + 1 - m| < 2p^{1/2}$, build an elliptic curve $E(F_p)$ such that $\#E(F_p) = m$.

The algorithm is based on constructing an elliptic curve $E(GF(2^m))$ that has a given j-invariant. However, the rest of the algorithm is based heavily on the algebraic number theory whose context is too complicated to be described in this document.

Refer to Müller & Paulus [MP97] and Baier & Buchmann [BB01] for more discussions on the CM methods. For CM methods and other works on Koblitz cureves, readers may refer to Koblitz ([K90],[K92]), Solinas ([S97],[S00]) and Meier & Staffelbach [MS93].

c. Comparisons

There is no mathematical theory, comparison or fact yet about which elliptic curves, randomly chosen or specially chosen, are more difficult for the ECDLP. There is current research mentioning general doubts about constructing elliptic curves extra internal structure, such as with special coefficients (i.e., Koblitz curves), with special Complex Multiplication property and/or over composite extension finite fields. There is a tradeoff between performance and security in implementing elliptic curve cryptosystems using such special curves and/or finite fields.

4.B.3. Finding a point of given prime order on an elliptic curve

The order n of a point $P \neq O$ on an elliptic curve is a positive integer such that

$$nP = O \text{ and } mP \neq O \text{ for any integer } m \text{ such that } 1 \le m < n.$$

The order n of a point must divide the order N of the elliptic curve. In fact, it is true for any group. If the elliptic curve order $N = \#E$ is a prime number, then the group is cyclic, and obviously all points except the point at infinity O are of order N.

Choosing a point P of prime order n: A simple method is usually applied in cryptographic practices when n is a large prime. Then the factor $l = \#E/n$ will not be divisible by n. Choose a random point $Q \neq O$ on the elliptic curve E, then verify whether the point $P = l \cdot Q$ has order n. This can be done simply by checking that $n \cdot P = O$. (Since n is prime, there is no other positive integer $m < n$ such that $m \cdot P = O$.) If it is true, then $P = l \cdot Q$ is the point we need; otherwise, choose another point Q and repeat.

4.B.4. Methods/formulae to compute the order of an elliptic curve

It should be noted that it is easy to check whether the number of points on an elliptic curve is correct when it is known, while an efficient algorithm to find out that number is still a difficult task.

In chapter 1, we already presented formulae and algorithms for counting the order of an elliptic curve group, such as Hasse-Weil theorem, direct formulae using Legendre symbol and trace function, Shanks' Baby-step-Giant-step algorithm and Schoof's algorithm.

a. The order of an elliptic curve of the following special forms

$$E_p(a, 0): y^2 = x^3 + ax, \text{ for } a \not\equiv 0 \pmod{p} \text{ and } p \equiv 1 \pmod{4}$$

or $E_p(0, b)$: $y^2 = x^3 + b$, for $b \not\equiv 0 \pmod p$ and $p \equiv 1 \pmod 3$,
over a prime finite field F_p are discussed thoroughly in Bressoud [B87] and Ireland & Rosen [IR90].

Case 1. For prime $p \equiv 1 \pmod 4$ and $E_p(a,0)$: $y^2 = x^3 + ax$, for $a \not\equiv 0 \pmod p$.

Let $r = s + it$ be a complex prime (where s and t are integers,) and let $\bar{r} = s - it$ be the complex conjugate of r, which satisfy the conditions $r \equiv 1 \pmod{2 + 2i}$ and $p = r\bar{r} = s^2 + t^2$. Let $\Re(r) = \Re(s + it) = s$ be the real part of number r. Then $\#E_p(a, 0) = p + 1 - 2\Re[R_4(-a)\,\bar{r}\,]$, where the symbol R_4 is defined by $R_4(x) = x^{(p-1)/4} \pmod r$.

The orders in this case are always even. Explicitly, the elliptic curve orders will fall into one of only four cases listed in this table.

If $R_4(-a) = (-a)^{(p-1)/4} \pmod r \equiv$	The order $\#E_p(a, 0)$ is
1	$p + 1 - 2\Re(\bar{r}) = p + 1 - 2s$
–1	$p + 1 - 2\Re(-\bar{r}) = p + 1 + 2s$
i	$p + 1 - 2\Re(i\bar{r}) = p + 1 - 2t$
$-i$	$p + 1 - 2\Re(-i\bar{r}) = p + 1 + 2t$

Table 4.4. Orders of elliptic curves $E_p(a,0)$: $y^2 = x^3 + ax$, $a \not\equiv 0 \pmod p$, over a prime finite field F_p, $p \equiv 1 \pmod 4$

Observe that $r \equiv 1 \pmod{2 + 2i}$ or $(2 + 2i) \mid (r - 1) = [(s - 1) + it]$. Hence we derive a relation on their squares of absolute values. It is: $8 \mid [(s - 1)^2 + t^2] = p + 1 - 2s$.

So we can solve the system of equations $s \equiv (p + 1)/2 \pmod 4$ and $p = s^2 + t^2$ for values s and t, such that $1 \le s,\ t < p^{1/2}$. Then choose and check an appropriate value of $r = \pm s \pm it \equiv 1 \bmod (2 + 2i)$.

Case 2. For prime $p \equiv 1 \pmod 3$ and $E_p(0, b)$: $y^2 = x^3 + b$ for $b \not\equiv 0 \pmod p$.

Let w be a non-trivial cubic root of 1, (i.e., $w^2 + w + 1 = 0$,) and $w = e^{2\pi i/3} = (-1 + i3^{1/2})/2$. Let $r = s + wt$ be a complex prime, where s and t are integers, satisfying the conditions $r \equiv 2 \pmod 3$ and $p = r\bar{r} = s^2 - st + t^2$. In this case, we note that $w^2 = (-1 - i3^{1/2})/2 = \bar{w}$ and $\bar{r} = (s - t) - wt$, for simpler calculations. Then $\#E_p(0, b) = p + 1 + 2\Re[R_6(4b)\,\bar{r}\,]$, where the symbol R_6 is defined by $R_6(x) = x^{(p-1)/6} \pmod r$.

Explicitly, the elliptic curve orders will fall into one of only six cases described in table 4.5.

If $R_6(b) = b^{(p-1)/6} \pmod r \equiv$	Equivalently, if $R_6(4b) = (4b)^{(p-1)/6} \pmod r \equiv$	The order $\#E_p(0,b)$ is
$u = 4^{(p-1)/3} \pmod r$	1	$p + 1 + 2\Re(\bar{r}) = p + 1 + 2s - t$
$-u$	–1	$p + 1 + 2\Re(-\bar{r}) = p + 1 - 2s + t$
$w.u$	w	$p + 1 + 2\Re(w\bar{r}) = p + 1 - s + 2t$
$-w.u$	$-w$	$p + 1 + 2\Re(-w\bar{r}) = p + 1 + s - 2t$
$w^2.u$	w^2	$p + 1 + 2\Re(w^2\bar{r}) = p + 1 - s - t$
$-w^2u$	$-w^2$	$p + 1 + 2\Re(-w^2\bar{r}) = p + 1 + s + t$

Table 4.5. Orders of elliptic curves $E_p(0, b)$: $y^2 = x^3 + b$, $b \not\equiv 0 \pmod p$ over a finite field F_p, $p \equiv 1 \pmod 3$

4.B.5. Schoof's and Satoh's algorithm for point-counting

Refer to chapter 1 for a short summary of development of the algorithms by many researchers such as Schoof, Atkin, Elkies, Couveignes and Lercier. We choose not to go

deeply into the theory behind these developments of the algorithms at the present time, because of the heavy burden of introducing a lot of difficult materials of the mathematics underlying.

We have discussed how to represent elements of finite fields and how to build an elliptic curve over the underlying finite fields for an elliptic curve cryptosystem. In the next section of this chapter, more features or issues on the implementations of operations of such cryptosystems will be discussed. They are scalar point multiplication formulae and algorithms, representations of points on an elliptic curve and special algorithms on Koblitz curves and composite extension finite fields and other topics.

4.C. Implementations of elliptic curve arithmetic operations

Many topics in implementations of arithmetic operations over elliptic curves will be discussed in this section: scalar point multiplications, methods representing points of an elliptic curve and Complex Multiplication methods…

The most basic operation is adding two points or doubling a point on an elliptic curve. It is more expensive computationally than a basic operation in a symmetric key cryptosystem (a block encryption/decryption). But it is still much faster than a basic modular multiplication over a cyclic group whose order is of the same security level.

We now discuss efficient algorithms to expedite implementation procedures in elliptic curve cryptosystems.

4.C.1. Scalar point multiplication: basic methods

One crucial operation is scalar point multiplication since it determines the speed of an elliptic curve cryptosystem. We will multiply a point P on an elliptic curve E by a positive integer k. By definition, $kP = \underbrace{P + P + \cdots + P}_{k \text{ terms}}$. This problem is analogous to raising an element to the k-th power in the multiplicative subgroup $GF(q)^*$.

a. Double-and-add method

This most basic method uses the binary expansion of the number k.

Let write $k = (k_{r-1},\ldots, k_0)$ in base 2, where $k_{r-1} = 1$ and $r = \lfloor \log_2 k \rfloor + 1$. Let $P_{r-2} = P$. Then compute $P_{i-1} = 2P_i + k_iP$, for all i, $r - 2 \geq i \geq 0$.

Then $kP = P_{-1} = 2P_0 + k_0P$.

This method requires $(r - 1)$ doublings and probabilistically about $(r - 1)/2$ additions or at most $(r - 1)$. Observe that we can reduce the number of arithmetic operations when the number of bits 0 is increased. This is the basic idea for methods, which try to improve the implementation of scalar point multiplication.

b. Addition-subtraction method

For elliptic curve implementation, the methods, which included subtractions, are more attractive than the corresponding methods, which included divisions in calculating power in finite fields. The reason is division or inversion in finite fields is a more costly operation than multiplication, while subtraction is just as costly as addition in elliptic curve operations.

The basic method uses the binary expansion of k and $3k$. Let $l = 3k = (l_{r-1},\ldots, l_0)$ and $k = (k_{r-1},\ldots, k_0)$ such that the leftmost bit l_{r-1} must be 1. That is, a few leftmost bits of k are added bits 0 from the canonical binary form of k.

Let $P_{r-1} = P$. Then compute $P_{i-1} = 2P_i + (l_{i-1} - k_{i-1})P$, for all i such that $r - 1 \geq i \geq 2$. Then $kP = P_1 = 2P_2 + (l_1 - k_1)P$.

This method requires $(r - 2)$ doublings and probabilistically about $(r - 2)/2$ additions/subtractions or at most $(r - 2)$. This method in fact is an easily described version of the following method.

c. Addition-subtraction method using NAF

We now use the canonical encoding called the non-adjacent form (NAF) of scalar k. This coding employs a signed binary expansion (using 0 and ±1) that has the property that no two consecutive coefficients are nonzero.

The NAF of an integer is unique and has the fewest nonzero coefficients of any signed binary expansions. There are many ways to construct the NAF. One way is just described previously, using bits in k and $3k$.

The NAF can be computed similarly by the same method for the binary form. That is, repeating dividing by 2 to collect the remainders, except an important rule: for the nonzero remainder, the corresponding quotient must be even. This exception helps to make the next remainder be zero.

Let k be in NAF, $k = (k_{r-1},\ldots, k_0)$. Let $P_{r-1} = P$. Then compute $P_{i-1} = 2P_i + k_{i-1}P$, for all i, $r - 1 \geq i \geq 1$. Then $kP = P_0 = 2P_1 + k_0P$.

This method requires $(r - 1)$ doublings and probabilistically about $(r - 1)/3$ additions or at most $\lceil (r - 1)/2 \rceil$.

Eventually, we can combine this method with the sliding window method for a more efficient implementation - the signed binary window method that will be discussed later.

d. m-ary method (or 2^d-ary method)

This method is generalized from the double-and-add method, where the m-ary expansion is used instead of binary form, where m is a power of 2. Let $m = 2^d$, where $d > 1$. We start with the binary expansion of the number $k = (k_{s-1},\ldots, k_0)$, where we may pad an extra number of bit 0's to the left side of the bit string to make $s = d.r$ for some integer r. Then we have the m-ary expansion of k of the form $k = (K_{r-1},\ldots, K_0)$ where each K_i is a d-bit string and $K_{r-1} \neq (0\ldots0)$.

First we pre-compute all points $2P$, $3P$,…, $(2^d - 1)P$. They will cover all possible points of the form K_iP. We look up for the point $P_{r-2} = K_{r-1}P$, and compute $P_{i-1} = 2^dP_i + K_iP$, for all i such that $r - 2 \geq i \geq 0$. Then $kP = P_{-1} = 2^dP_0 + K_0P$.

This method requires $(2^d - 2)$ pre-computations (and memory storage), $(r - 1)d = s - d$ doublings and probabilistically about $(r - 1)(1 - 2^{-d})$ additions. We can observe that the larger d is, the more pre-computations are needed. With a little calculus, we can find the optimal d to minimize the total of additions: $A(s, d) = 2^d - 2 + s - d + [(s/d) - 1] (1 - 2^{-d})$.

Bit recoding techniques, such as signed binary expansions, are also used to improve the binary or m-ary methods. Refer to Koç [Kc91] and Eğecioğlu & Koç [EK94] for detailed analyses in this approach.

e. The 2^d-ary NAF form

It is possible that one can combine the addition-subtraction method with the 2^d-ary method. Particularly, the addition-subtraction method using the 2^d-ary NAF form that is a binary form with the property that there is at most one non-zero term in d consecutive coefficients. This form always uniquely exists and is easy to compute. The computing method is similar to that for the NAF form, except that the corresponding quotient to the

non-zero remainder must be divisible by 2^{d-1}. The pre-computation must store all points: $\pm P, \pm 3P, \ldots, \pm(2^{d-1} - 1)P$.

The addition-subtraction method is a special case for $d = 2$.

4.C.2. Scalar point multiplication: advanced methods

a. Sliding window method ([Kc95])

This method aims to separate zero words so we can skip an addition in the m-ary method discussed above. Instead of decomposing $k = (k_{s-1}, \ldots, k_0)$ into words of d-bit length, we now decompose k into zero and nonzero words, or windows W_i of varying lengths l_i. Let d be the maximum length of all nonzero windows. Then we need to pre-compute only "odd scalar multiplying" points $3P, 5P, \ldots, (2^d - 1)P$ We write $k = (W_{r-1}, \ldots, W_0)$ where W_{r-1} is a non-zero window (or window number).

Look up for $P_{r-2} = W_{r-1}P$. Then compute $P_{i-1} = 2^{l_i} P_i + W_i P$, for all i such that $r - 2 \geq i \geq 0$. Then $kP = P_{-1} = 2^{l_0} P_0 + W_0 P$.

There are two strategies to partition a binary expansion into windows: constant length and variable length nonzero windows.

Constant length nonzero windows

This strategy tries to produce zero windows of arbitrary length and nonzero windows of a fixed length d. A nonzero window will start when a bit 1 is encountered as we scan the bits from rightmost bit to leftmost.

This method requires $(2^d - 2)/2 = 2^{d-1} - 1$ pre-computations (and memory storage), $(s - d)$ doublings and probabilistically about A additions, where A is the number of non-zero windows, $A \leq \lceil s/d \rceil$. Refer to Koç [Kc95] for more analytic results on the value of A. In summary, this method reduces the number of additions by 3 to 7%, for $128 \leq s \leq 2048$, less than the m-ary method.

Variable length nonzero windows

This strategy tries to produce nonzero windows whose right-end and left-end bits are both 1. Two parameters are to be decided: the maximum length d of nonzero windows and the maximum number r of adjacent 0's allowed inside any nonzero window.

This method generally tries to decrease further the average number of nonzero windows when d and r are chosen optimally. We should choose $4 \leq d \leq 8$.

This method requires $(2^d - 2)/2 = 2^{d-1} - 1$ pre-computations (and memory storage), probabilistically about A additions, where A is the number of non-zero windows, $A \leq \lceil s/d \rceil$ and D doublings. Refer to Koç [Kc95] for more analytic results on the values of A and D. In summary, this method reduces the number of additions by 5-8%, for $128 \leq s \leq 2048$, less than the m-ary method.

b. Signed binary window methods

These methods transform an ordinary binary expansion B into a signed binary expansion S. (The transformations are also called "bit recoding" techniques.) Again, the methods or algorithms using "signed" expansion are much more efficient in implementing elliptic curve operations than in finite field operations since the subtraction is just as costly as addition.

The purpose is to skip a bit string of 1's (in addition to bit strings of 0's, as usual) to reduce the number of additions.

Morain-Olivos' algorithm

This algorithm reduces the weight of the signed binary form S, i.e., the number of non-zero digits, denoted by $\#_1(S)$. The idea is that a block of n bits 1 can be replaced by a bit string that is a block consisting of a bit 1 followed by n bits 0 and then minus 1. That is: $\underbrace{11\cdots1}_{n \text{ bits}} = 1\underbrace{00\cdots0}_{n \text{ bits}} - 1$. This observation is extracted from the equality:

$$2^{n+1} - 1 = (2^n + 2^{n-1} + \ldots + 2^1 + 2^0).$$

As a result, $(n - 1)$ doublings and $(n - 1)$ additions (i.e., $2(n - 1)$ total additions) can be replaced by n doublings and 1 subtraction (i.e., $(n + 1)$ total additions). In other words, this method tries to construct two positive integers k_+ and k_- such that $k_+ - k_- = k$. The total computation for $(k_+ - k_-).P$ is less than that for kP. Note that there are not two separate computations of k_+P and k_-P, but actually the computations merge together: k_-P only shows up in a few subtractions corresponding to the positions of its bits 1 in the scalar k.

The same idea is to deal with a special string that has isolated 0's. We observe that: $k = \underbrace{1\cdots1}_{n \text{ bits}}0\underbrace{1\cdots1}_{m \text{ bits}} = 1\underbrace{0\cdots0}_{n \text{ bits}}\bar{1}\underbrace{0\cdots\cdots0}_{(m-1) \text{ bits}}\bar{1} = 1\underbrace{0\cdots0}_{n+m+1 \text{ bits}} - 0\underbrace{0\cdots0}_{n \text{ bits}}1\underbrace{0\cdots\cdots0}_{(m-1) \text{ bits}}1$, where bit $\bar{1}$ denotes (-1), and we assume $m \geq 2$. This observation is extracted from the same equality above, applied twice. Then we have the formula: $kP = 2^m.(2^{n+1}.P - P) - P$. That is, an isolated 0 inside a block of bit 1's will contribute only two subtractions/additions and one extra doubling, instead of $(n + m - 1)$ additions.

Morain-Olivos [MO90] provides detailed estimations of implementation cost. In summary, the method reduces about 3% for 100-digit number and 2.7% for 300-digit number.

Jedwab & Mitchell [JM89] also proposed similar approaches using a modified signed-digit representation. The original idea was proposed by Mitchell & Selby [MS89]. Müller [Mu98] discussed improved versions over Morain-Olivos' method.

A generalization of the Morain-Olivos' algorithm

We can generalize this result for k being a string of $(b - 1)$ isolated bits 0 sandwiched among b blocks of bits 1, where b is larger than 2, the following can be written: $k = \underbrace{1\cdots\cdots1}_{N_1 \text{ bits}}0\cdots\underbrace{1\cdots\cdots1}_{N_{b-1} \text{ bits}}0\underbrace{1\cdots\cdots1}_{N_b \text{ bits}} = 1\underbrace{0\cdots\cdots\cdots\cdots0}_{N_1+\cdots+N_b+(b-1) \text{ bits}} - 0\underbrace{0\cdots0}_{N_1 \text{ bits}}1\cdots\underbrace{0\cdots\cdots0}_{N_{b-1} \text{ bits}}1\underbrace{0\cdots\cdots0}_{(N_b-1) \text{ bits}}1,$

assuming $N_b \geq 2$. This observation is also derived from the above equality:

$$2^{n+1} - 1 = (2^n + 2^{n-1} + \ldots + 2^1 + 2^0).$$

Then we can obtain the following formula: $kP = 2^{N_b}\cdot(2^{N_{b-1}+1}\cdot[\cdots(2^{N_1+1}\cdot P - P)\cdots] - P) - P$.

This formula dramatically generalized the application of Morain-Olivos' algorithm such that b can be any positive number greater than 2, rather than being restricted to $b = 2$ only.

By applying this algorithm, $(N_1 + \ldots + N_b - 1)$ additions were replaced by only b subtractions/additions and one extra doubling. The savings in the number of arithmetic operations are significant when the sum $(N_1 + \ldots + N_b)$ is much larger than b, which should be obtainable for those cases of k.

Koyama-Tsuruoka's algorithm ([KT92])

This algorithm improved the above methods by increasing the average length of zeros in the signed binary expansion using $\{\bar{1}, 0, 1\}$, where bit $\bar{1}$ denotes (-1).

A binary string of a non-zero window $B = (1, b_n,\ldots, b_i,\ldots, b_1, 1)$ in k will be transformed to a signed binary string of the form $T = (1, 0, t_n,\ldots, t_i,\ldots, t_1, \bar{1})$, where $t_i = b_i - 1$, for all $1 \le i \le n$. This transformation is effective (i.e. actually decreases the weight of the bit string) only when the difference between the numbers of bits 1 and bits 0 is: $\mathrm{Diff}(B) = \#_1(B) - \#_0(B) > 2$. However, we should keep in mind that the transformation also costs us one extra doubling because of the extra bit.

Both methods (by Koyama-Tsuruoka and Morain-Olivos) generate a signed bit string with the same weight, but the average length of zero runs by the Koyama-Tsuruoka method is greater than that of the Morain-Olivos method.

By this method, one needs to pre-compute only the odd scalar points $\pm 3P$, $\pm 5P$,…, upto $\pm(2^d - 3)P$, since this algorithm never allows the points $\pm(2^d - 1)P$ to appear.

In fact, this transformation is extracted from the equality:

$$2^{n+2} - 2^{n+1} = 2^{n+1} = (2^n + 2^{n-1} + \ldots + 2^1 + 1 + 1).$$

Using the relationship $1 = b_i - t_i$, for all $1 \le i \le n$, we then have:

$$2^{n+2} + (t_n 2^n + t_{n-1}2^{n-1} + \ldots + t_1 2^1 + 1) = 2^{n+1} + (b_n 2^n + b_{n-1}2^{n-1} + \ldots + b_1 2^1 + 1).$$

Refer to Koyama & Tsuruoka [KT92] for detailed performance evaluation and comparison. Note that, the similar idea in the Koyama-Tsuruoka's algorithm was also discussed in Koç [Kc91].

A generalization of Koyama-Tsuruoka's algorithm

Instead of applying this algorithm for a non-zero window only, we try to apply for an arbitrary bit string $k = (b_n,\ldots, b_1, b_0)$, where without loss of generality, we may assume $b_n = 1$. For all $0 \le i \le n$, let $t_i = b_i - 1$, then again insert it into the arithmetic identity: $2^{n+1} = (2^n + 2^{n-1} + \ldots + 2^1 + 2^0) + 1$, we can obtain the following identity:

$$2^{n+1} + (t_n 2^n + t_{n-1}2^{n-1} + \ldots + t_1 2^1) + t_0 2^0 - 1) = (b_n 2^n + b_{n-1}2^{n-1} + \ldots + b_1 2^1 + b_0 2^0).$$

When $b_0 = 1$, then $t_0 = 0$, this approach will transform k to the string of digits $T = (1, t_n,\ldots, t_1, \bar{1})$, where the last digit $\bar{1} = -1$. This is Koyama-Tsuruoka's algorithm for non-zero window $B = (b_n,\ldots, b_1, 1)$.

When $b_0 = 0$, then $t_0 = -1$, this approach will transform $k = (b_n,\ldots, b_1, 0)$, to the string of digits $T = (1, t_n,\ldots, t_1, \bar{2})$, where the last digit $\bar{2} = -2$. This last digit does not affect the scalar point multiplication $(k \cdot P)$ at all. In the very last step of a given scalar point multiplication algorithm, we then subtract a double of a point, $2P$, instead of the point P itself. The point $2P$ is available for free since it is always computed during the process. If we use other m-ary methods or window methods, the digit $\bar{2}$ obviously is not a concern anyway. We need only minor changes in pre-computations.

The number of non-zero digits in T is

$$\#_{\text{non-0}}(T) = 2 + \sum_{i=1}^{n} |t_i| = 2 + \sum_{i=1}^{n} |b_i - 1| = 2 + \sum_{1 \le i \le n, b_i = 0} (1) = 2 + \#_0(k'), \text{ where } k' = (b_n,\ldots, b_1).$$

Hence this transformation is effective if the condition $2 + \#_0(k') < \#_1(k)$ is satisfied. Since $\#_0(k') = \#_0(k) - 1$, we can rewrite the condition as:

$$\mathrm{Diff}(k) = \#_1(k) - \#_0(k) > 1.$$

Hence Koyama-Tsuruoka's algorithm can extend for any bit strings, which satisfy this condition. Again, the transformation also costs us one extra doubling because of the extra bit.

c. Other algorithms/discussions

Jedwab & Mitchell [JM89] proposed an equivalent algorithm to Morain-Olivos' method. The original idea was proposed by Mitchell & Selby [MS89]. Müller [Mu98] discussed improved versions over Morain-Olivos' method.

All algorithms described above are based on sliding window and bit manipulations. They gave almost the same level of performance. They are all doing better than a typical *m*-ary method. However, no method actually dominates over the other methods yet. There is no significant difference or breakthrough in the algorithms showed above.

The attractive point again is using "negative" digits (e.g., $\overline{1} = -1$ and/or $\overline{2} = -2$) to have the subtractions involved since the cost of subtraction is the same as that of addition in elliptic curve implementation.

Refer to Gollmann, Han & Mitchell [GHM96], Brickell, Gordon, McCurley & Wilson [BGMW93] and Gordon [G98] for more discussions.

d. Methods using addition chains/sequences

The problem of optimal addition chains is to find the fewest additions needed to compute a positive integer k starting from 1. It is used to compute kP from P with the fewest elliptic curve additions (originally, to compute the power x^k from x with fewest multiplications).

An *addition chain* $[A_i]$ of k of length L is of the form: $1 = A_0 < A_1 < \ldots < A_L = k$, where every number is the sum of two earlier numbers: For $1 \le i \le L$, $A_i = A_j + A_m$, where $i > j \ge m$.

An *addition sequence* of k is of the form: $1 = A_0 < A_1 < \ldots < A_L = k < A_{L+1} < \ldots$, where every number is the sum of two earlier numbers as in an addition chain. Note that in an addition chain, k occurs at the very end of the chain, while in an addition sequence, k just needs to occur someplace in the sequence. For our practical purpose, now we mention only addition chains.

Obviously, there can be many different addition chains for a given positive integer k. Naturally, we are most interested in finding the addition chain of minimum length since it will help to minimize the number of arithmetic operations.

In another version, we call an *addition/subtraction chain*, when $[A_i]$ satisfies weaker conditions: for $1 \le i \le L$, $A_i = A_j + A_m$ or $A_i = A_j - A_m$, where $i > j \ge m$.

In elliptic curve implementation, the subtraction operation is just as costly as addition; the addition/subtraction chains would be more attractive here than they are in an exponentiation problem.

A *star chain* is an addition chain $[A_i]$ satisfying: for $1 \le i \le L$, $A_i = 2A_{i-1}$ (a doubling) or $A_i = A_{i-1} + A_l$, ("star step") where $i - 1 > l$. That is, one summand must be the very previous element. A star step is called a "simple step" when $l = 1$.

Refer to Downey, Leong & Ravi Sethi [DLS81], Volger [V85] and Schönhage [S75], van Leeuwen [L77], McCarthy [Mc86], Dietel & Sauerbrey [DS92] and de Rooij's [R98] for more discussions.

Generally, computing the addition chain of minimum length is a very difficult problem, but there are simple algorithms to produce good addition chains, even near minimum length. The most common algorithm is using the binary form as in double-and-add (or addition-subtraction) method, even though it usually produces addition chain

larger than minimum length. Finding the addition chain with the minimum length will benefit the algorithms to compute the scalar point multiplication kP, since it will minimize the number of required additions.

Bos & Coster [BC90] discussed a heuristic routine to create an addition sequence of a set of numbers. This method will be significant when sliding window methods of large width are used. In those cases, the pre-computed table will require a large number of computations and also storage. We do not even need the addition chain for k. Instead, we need to create only an addition sequence that consists of the needed window numbers only. These window numbers are of course less than 2^w, where w is the window width. Hence they are much smaller than k itself. Then using one of the methods discussed previously, we can create the addition chain for k.

Yacobi [Y91] proposed a similar method, systematically developing into a heuristic algorithm (that is claimed to be a modification of the Lempel-Ziv data compression algorithm.) We will apply a 2^d-ary method (or sliding window method) where we will pre-compute only those intermediate scalar point multiplications that will be needed. However, this algorithm is not better than a 2^d-ary method for small scalar k, less than 512 bits, but it is more efficient for larger k.

Most recently, Aigner & Oswald [AO01] discussed using randomized addition-subtraction chains in counter-attacks against differential power attacks.

4.C.3. Scalar point multiplication: other methods

a. Using projective coordinates

Menezes & Vanstone ([MV90],[MV93]) proposed ideas of using projective coordinates in implementing scalar point multiplication. It is to remove the inversion operations in point addition or doubling operations in the intermediate steps in any scalar point multiplication algorithm implemented. At the final step, we can use a single inversion to convert it to affine coordinates as usual.

Menezes & Vanstone ([MV90],[MV93]) worked on the binary finite field $GF(2^m)$. Koyama & Tsuruoka [KT92] worked over prime finite field F_p They were all dealing with the usual projective coordinates (xz, yz, z).

Let $P = (x_1, y_1, 1)$ and $Q = (x_2, y_2, z_2)$. We can rewrite $Q = (x_2 z_2^{-1}, y_2 z_2^{-1}, 1)$ and apply the regular point addition formulae (for affine coordinates) to find $R = P + Q = (x'_3, y'_3, 1)$. Then let z_3 be the common denominators of x'_3 and y'_3, we can write $R = (x_3, y_3, z_3)$. The completed results for all three usual cases of elliptic curves over finite fields are summarized in the table 4.6. For convenient reference, we also include the formulae of the additive inverse of a point in the first column.

Equation of elliptic curve E over F_q in affine coordinates	$R = (x_3, y_3, z_3) = (x_1, y_1, 1) + (x_2, y_2, z_2) = P + Q$ Let $A = x_1 z_2 + x_2$, $B = y_1 z_2 + y_2$, $X = x_2 - x_1 z_2$, $Y = y_2 - y_1 z_2$
Over F_p, $p \neq 2, 3$ $y^2 = x^3 + ax + b$ $\Delta = -16(4a^3 + 27b^2)$ $\Delta \neq 0$ $P = (x_1, y_1, 1)$ $Q = (x_2, y_2, z_2)$ Then	$x_3 = \begin{cases} XT & \text{if } P \neq \pm Q \\ 2y_1(S^2 - 8x_1 y_1^2) & \text{if } P = Q, \end{cases}$ $y_3 = \begin{cases} (x_1 y_2 - x_2 y_1) X^2 z_2 - YT & \text{if } P \neq \pm Q \\ S(12x_1 y_1^2 - S^2) - 8y_1^4 & \text{if } P = Q, \end{cases}$

$-Q = (x_2, -y_2, z_2)$	$z_3 = \begin{cases} X^3 z_2 & \text{if } P \neq \pm Q \\ 8y_1^3 & \text{if } P = Q, \end{cases}$ where $T = Y^2 z_2 - AX^2$ and $S = 3x_1^2 + a$. (A slightly revised version of Koyama & Tsuruoka)
Non-supersingular elliptic curve over $GF(2^m)$ $y^2 + xy = x^3 + ax^2 + b$ $b \neq 0$ $P = (x_1, y_1, 1)$ $Q = (x_2, y_2, z_2)$ Then $-Q = (x_2, y_2 + x_2, z_2)$	$x_3 = \begin{cases} AV & \text{if } P \neq \pm Q \\ Ux_1 & \text{if } P = Q, \end{cases}$ $y_3 = \begin{cases} A^2 z_2 (x_1 y_2 + x_2 y_1) + V(A + B) & \text{if } P \neq \pm Q \\ U(x_1^2 + y_1) + bx_1 & \text{if } P = Q, \end{cases}$ $z_3 = \begin{cases} A^3 z_2 & \text{if } P \neq \pm Q \\ x_1^3 & \text{if } P = Q, \end{cases}$ where $V = A^2(A + az_2) + Bz_2(A + B)$ and $U = x_1^4 + b$ (A slightly revised version of Menezes & Vanstone)
Supersingular elliptic curve over $GF(2^m)$ $y^2 + cy = x^3 + ax + b$ $c \neq 0$ $P = (x_1, y_1, 1)$ $Q = (x_2, y_2, z_2)$ Then $-Q = (x_2, y_2 + cz_2, z_2)$	$x_3 = \begin{cases} AW & \text{if } P \neq \pm Q \\ cZ^2 & \text{if } P = Q, \end{cases}$ $y_3 = \begin{cases} B(B^2 z_2 + A^2 x_2) + (y_1 + c) z_3 & \text{if } P \neq \pm Q \\ Z(x_1 c^2 + Z^2) + z_3 (y_1 + c) & \text{if } P = Q, \end{cases}$ $z_3 = \begin{cases} A^3 z_2 & \text{if } P \neq \pm Q \\ c^3 & \text{if } P = Q, \end{cases}$ where $W = B^2 z_2 + A^3$ and $Z = x_1^2 + a$ (Menezes & Vanstone)

Table 4.6. Point addition formulae in the projective coordinates (xz, yz, z)

With the above formulae, we can compute the scalar point multiplication, $kP = k.(x_1, y_1, 1) = (x, y, z)$, by repeated additions, as usual. Then by a single inversion at the very end step to get z^{-1}, we can write $kP = (xz^{-1}, yz^{-1})$. The disadvantage of this method is that it requires larger memory to store points of three coordinates on an elliptic curve.

We can develop other scalar point multiplication formulae by using the Jacobian projective coordinates, (z^2x, z^3y, z) , $z \neq 0$. Again, let $P = (x_1, y_1, 1)$ and $Q = (x_2, y_2, z_2)$. We rewrite $Q = (x_2 z_2^{-2}, y_2 z_2^{-3}, 1)$ and apply the regular addition formulae (for affine coordinates) to find $R = P + Q = (x'_3, y'_3, 1)$. Then we convert it back to the original Jacobian projective coordinates. The completed results for all three usual cases of elliptic curves over finite fields are summarized in the table 4.7.

Equation of elliptic curve E over F_q in affine coordinates	$R = (x_3, y_3, z_3) = (x_1, y_1, 1) + (x_2, y_2, z_2) = P + Q$ --- Let $X = x_1 z_2^2 + x_2, Y = y_1 z_2^3 + y_2, C = x_2 - x_1 z_2^2, D = y_2 - y_1 z_2^3$
Over F_p, $p \neq 2, 3$ $y^2 = x^3 + ax + b$ $\Delta = -16(4a^3 + 27b^2) \neq 0$ $P = (x_1, y_1, 1)$ $Q = (x_2, y_2, z_2)$	$x_3 = \begin{cases} D^2 - XC^2 & \text{if } P \neq \pm Q \\ S^2 - 8x_1 y_1^2 & \text{if } P = Q, \end{cases}$

Then $-Q = (x_2, -y_2, z_2)$	$y_3 = \begin{cases} D(x_1 z_3^2 - x_3) - y_1 z_3^3 & \text{if } P \neq \pm Q \\ S(4x_1 y_1^2 - x_3) - y_1 z_3^3 & \text{if } P = Q, \end{cases}$ $z_3 = \begin{cases} Cz_2 & \text{if } P \neq \pm Q \\ 2y_1 & \text{if } P = Q, \end{cases}$ where $S = 3x_1^2 + a$. (A slightly revised version of Cohen, Miyaji & Ono [CMO97])
Non-supersingular elliptic curve over $GF(2^m)$ $y^2 + xy = x^3 + ax^2 + b$ $b \neq 0$ $P = (x_1, y_1, 1)$ $Q = (x_2, y_2, z_2)$ Then $-Q = (x_2, y_2 + x_2 z_2, z_2)$	$x_3 = \begin{cases} X^2(az_2^2 + X) + YV & \text{if } P \neq \pm Q \\ U & \text{if } P = Q, \end{cases}$ $y_3 = \begin{cases} x_3 V + x_1 Y z_3^2 + y_1 z_3^3 & \text{if } P \neq \pm Q \\ U(x_1^2 + y_1) + bx_1 & \text{if } P = Q, \end{cases}$ $z_3 = \begin{cases} z_2 X & \text{if } P \neq \pm Q \\ x_1 & \text{if } P = Q, \end{cases}$ where $V = Xz_2 + Y$ and $U = x_1^4 + b$
Supersingular elliptic curve over $GF(2^m)$ $y^2 + cy = x^3 + ax + b$ $c \neq 0$ $P = (x_1, y_1, 1)$ $Q = (x_2, y_2, z_2)$ Then $-Q = (x_2, y_2 + cz_2^3, z_2)$	$x_3 = \begin{cases} Y^2 + X^3 & \text{if } P \neq \pm Q \\ Z^2 & \text{if } P = Q, \end{cases}$ $y_3 = \begin{cases} Y(Y^2 + X^2 x_2) + (y_1 + c)z_3^3 & \text{if } P \neq \pm Q \\ Z(c^2 x_1 + x_3) + c^3(y_1 + c) & \text{if } P = Q, \end{cases}$ $z_3 = \begin{cases} z_2 X & \text{if } P \neq \pm Q \\ c & \text{if } P = Q, \end{cases}$ where $Z = x_1^2 + a$

Table 4.7. Point addition formulae in the Jacobian projective coordinates (z^2x, z^3y, z)

Refer to Agnew, Mullin & Vanstone [AMV93] and Cohen, Miyaji & Ono ([CMO97], [CMO98]) for more discussions.

b. Montgomery's method

This method is extracted from a work of Montgomery [M87]. In the table 4.8., we show the relations of the x-coordinates of 2 points $(P + Q)$ and $(P - Q)$.

Equation of elliptic curve E over finite field	$P + Q = (x_3, y_3)$ & $P - Q = (x_4, y_4)$, where $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and $P \neq \pm Q$
Over F_p, $P \neq 2, 3$, E: $y^2 = x^3 + Ax + B$ $\Delta = -16(4a^3 + 27b^2) \neq 0$)	$x_3 = x_4 - 4y_1 y_2 (x_2 - x_1)^{-2}$
Non-supersingular elliptic curve over $GF(2^m)$ $y^2 + xy = x^3 + ax^2 + b$ (and $b \neq 0$)	$x_3 = x_4 + x_1 x_2 (x_1 + x_2)^{-2}$
Supersingular elliptic curve over $GF(2^m)$ $y^2 + cy = x^3 + ax + b$ (and $c \neq 0$)	$x_3 = x_4 + c^2 (x_1 + x_2)^{-2}$

Table 4.8. Addition formulae using the Montgomery's method

Hence, for both supersingular and non-supersingular elliptic curves over binary finite field $GF(2^m)$, we can compute the x-coordinate x_3 of $(P + Q)$ with only one

inversion and a few finite field additions from the x-coordinates of P, Q and $(P - Q)$. The y-coordinates are not involved in the computations.

However, if over prime finite field F_p, the computation must involve also the y-coordinates y_1 and y_2. Hence the Montgomery's method performs better over binary finite fields than over prime finite fields.

To compute point kP, where $k = (k_{r-1},\ldots, k_0)$ in base 2, we first compute the point $2P$. Thereafter, given a pair of points $(mP, (m + 1)\cdot P)$, we then compute in step i either

$$(2m\cdot P, (2m + 1)\cdot P) \quad \text{if } k_i = 0, \text{ or}$$
$$((2m + 1)\cdot P, (2m + 2)\cdot P) \quad \text{if } k_i = 1.$$

The Montgomery's method has a considerably slower speed since in each step (for each bit) we must compute

A doubling in order to get the point $2m\cdot P$ or $(2m + 2)\cdot P = 2(m + 1)\cdot P$) and

A point addition with point $(\pm P)$ to get $(2m + 1)\cdot P$.

On the other hand, these computations also provide one advantage of this method: the ability of resistance against the power differential analyses attacks since there is no distinction on operating over bit 0 or bit 1.

One may even use projective coordinates in the Montgomery's method to reduce the inversion (or division). We summarize the formulae in the table 4.9.

Equation of elliptic curve E over finite field F_q	$P + Q = (x_3, y_3, z_3)$ & $P - Q = (x_4, y_4, z_4)$, where $P = (x_1, y_1, 1)$, $Q = (x_2, y_2, z_2)$ and $P \neq \pm Q$
Over F_p, $p \neq 2, 3$ $y^2 = x^3 + ax + b$ (and $\Delta = -16(4a^3 + 27b^2) \neq 0$)	$x_3 = x_4 - 4 z_2^2 y_1 y_2 (x_2 - x_1 z_2)$ $z_3 = z_4$
Non-supersingular elliptic curve over $GF(2^m)$ $y^2 + xy = x^3 + ax^2 + b$ (and $b \neq 0$)	$x_3 = x_4 + z_2^2 x_1 x_2 (x_1 z_2 + x_2)$ $z_3 = z_4$ (Agnew, Mullin & Vanstone)
Supersingular elliptic curve over $GF(2^m)$ $y^2 + cy = x^3 + ax + b$ (and $c \neq 0$)	$x_3 = x_4 + z_2^3 c^2 (x_1 z_2 + x_2)$ $z_3 = z_4$

Table 4.9. Addition formulae using the Montgomery's method in the projective coordinates (xz, yz, z)

Hence, over binary finite fields $GF(2^m)$, for both supersingular and non-supersingular elliptic curves, we can compute the x-coordinate x_3 of $(P + Q)$ with a few field operations from z_2 and x-coordinates of three points P, Q and $(P - Q)$. Over prime finite fields F_p, the computation must involve also the y-coordinates y_1 and y_2. Moreover, no inversion or division is required.

We can do similar computations over the Jacobian projective coordinates of the form (z^2x, z^3y, z) , $z \neq 0$.

Equation of elliptic curve E over finite field F_q	$P + Q = (x_3, y_3, z_3)$ and $P - Q = (x_4, y_4, z_4)$, where $P = (x_1, y_1, 1)$, $Q = (x_2, y_2, z_2)$ and $P \neq \pm Q$
Over F_p, $p \neq 2, 3$ $y^2 = x^3 + ax + b$ (and $\Delta = -16(4a^3 + 27b^2) \neq 0$)	$x_3 = x_4 - 4 y_1 y_2 z_2^3$ $z_3 = z_4$

Non-supersingular elliptic curve over $GF(2^m)$ $y^2 + xy = x^3 + ax^2 + b$ (and $b \neq 0$)	$x_3 = x_4 + z_2^4 x_1 x_2$ $z_3 = z_4$
Supersingular elliptic curve over $GF(2^m)$ $y^2 + cy = x^3 + ax + b$ (and $c \neq 0$)	$x_3 = x_4 + z_2^6 c^2$ $z_3 = z_4$

Table 4.10. Addition formulae using the Montgomery's method in the Jacobian projective coordinates (z^2x, z^3y, z)

Again, over binary finite fields $GF(2^m)$, for both supersingular and non-supersingular elliptic curves, we can compute the x-coordinate x_3 of $(P + Q)$ with a few field operations from z_2 and x-coordinates of three points P, Q and $(P - Q)$. Over prime finite fields F_p, the computation must involve also y-coordinates y_1 and y_2. Moreover, no inversion or division is required.

In summary, the Montgomery's method always works better over binary finite fields than over prime finite fields for either affine coordinates or projective coordinates.

c. Demytko's work ([D94])

Let $P = (x, y)$ be a point on an elliptic curve E: $y^2 = x^3 + ax + b$ over finite field F_p with its discriminant $\Delta = -16(4a^3 + 27b^2) \neq 0$. Let $P_k = k.P = (x_k, y_k)$.

If $y_k \not\equiv 0 \pmod p$, then $x_{2k} = \dfrac{(x_k^2 - a)^2 - 8bx_k}{4y_k^2} = \dfrac{(x_k^2 - a)^2 - 8bx_k}{4(x_k^3 + ax_k + b)}$.

If $x_k \not\equiv x_{k+1}$ and $x \not\equiv 0 \pmod p$, then $x_{2k+1} = \dfrac{(a - x_k x_{k+1})^2 - 4b(x_k + x_{k+1})}{x(x_k - x_{k+1})^2}$.

If $x_k \not\equiv x_{k+1}$ and $x \equiv 0 \pmod p$, then $x_{2k+1} = \dfrac{4b + 2(a - x_k x_{k+1})(x_k + x_{k+1})}{(x_k - x_{k+1})^2} + x$.

We can observe that $x_k \equiv x_{k+1} \pmod p$ only when $P_k = -P_{k+1}$ or $(2k + 1)\cdot P = \mathrm{O}$.

Recall from the Montgomery's method, we have a more general formula

$$\text{If } x_i \not\equiv x_j \text{ then } x_{i+j} = \frac{4b + 2(a - x_i x_j)(x_i + x_j)}{(x_i - x_j)^2} + x_{i-j}.$$

This yields a chosen message attack proposed by Kaliski [K97].

We can also use the projective coordinates in the above formulae. For example, we denote $P = (X, Y, Z)$ and $P_k = k.P = (X_k, Y_k, Z_k)$. Then we have:

$X_{2k} = (X_k^2 - aZ_k^2)^2 - 8bX_k Z_k^3$; $Z_{2k} = 4Z_k(X_k^3 + aX_k Z_k^2 + bZ_k^3)$ and

$X_{2k+1} = Z.[(X_k X_{k+1} - aZ_k Z_{k+1})^2 - 4bZ_k Z_{k+1}(X_k Z_{k+1} + X_{k+1} Z_k)]$; $Z_{2k+1} = X(X_k Z_{k+1} - X_{k+1} Z_k)^2$.

d. Direct multiplication formulae and others

For practical implementation, the inversion of finite field elements is the most expensive operation to perform in finite fields. Guajardo & Paar proposed an idea to reduce the number of inversions at the cost of extra multiplications for calculating the point $(2^d.P)$. Instead of repeating doubling P many times to compute the intermediate points $2P$, 2^2P,…,$2^{d-1}P$, which may be of no use at all, we should derive a general direct formulae to compute point $(2^d.P)$ for any positive integer d, as large as one can. In each formula, we try to reduce the number of inversions to a possible minimum. Those formulae can be applied in the pre-computations of 2^d-ary methods or window methods to improve the efficiency.

Shamir's speed-up algorithm to compute point ($aP + bQ$) is to go through the non-adjacent signed binary expansions of the scalars a and b at the same time, doubling and adding/subtracting P, Q, and $P \pm Q$. This algorithm is claimed to have fast speed but it also requires more memory.

Others, such as Lee & Lim [LL94], and Brickell, Gordon, McCurley & Wilson [BGMW93] worked on pre-computations to improve scalar multiplication efficiency. Gallant, Lambert & Vanstone [GLV01] and Müller ([Mu98,[Mu98a]) discussed using efficient endomorphisms of elliptic curves.

4.C.4. Algorithms on composite extension finite fields

For a composite extension finite field, we mean a finite field of the form $GF(2^r)$, where r is a composite number $r = nm$. Then the finite field $GF(2^{mn})$ is considered an etension field of order m of the subfield $GF(2^n)$, (or an extension field of order n of the subfield $GF(2^m)$). Refer to Green & Taylor [GT74].

a. Multiplicative inversion over composite extension finite fields

Guajardo [G97], Paar [P95], Guajardo & Paar ([GP97],[GP98], [GP01]) aimed to generalize the generalized the Itoh-Tsujii algorithm (which was proposed for mornal bases) to polynomial bases and for composite extension fields.

It takes advantage of calculations in the subfield $GF(2^n)$ of small degree n. The reduction polynomial of $GF(2^{nm})$ over $GF(2^n)$ is $P(x)$. The inverse of a non-zero element $A \in GF(2^{nm})$ is defined by: $A^{-1} = (A^r)^{-1}A^{r-1}$, where $r = (2^{nm} - 1)/(2^n - 1)$ and $A^r \in GF(2^n)$. First, the term A^{r-1} will be computed using addition chains since $r - 1 = 2^n + 2^{2n} + \ldots + 2^{(m-1)n}$. Second, we observe that the product A^r of two elements A and A^{r-1} in $GF(2^{nm})$ is, in fact, in the subfield $GF(2^n)$. This helps to reduce the cost in comparison with general multiplication in the composite extension finite field $GF(2^{mn})$ if we choose the reduction polynomial $P(x)$ carefully. Third, the inversion $(A^r)^{-1}$ is easily performed in the subfield $GF(2^n)$. The final product between an element $(A^r)^{-1} \in GF(2^n)$ and an element $A^{r-1} \in GF(2^{nm})$ also requires only m multiplications in $GF(2^n)$ and no reduction modulo polynomial $P(x)$.

Fan & Paar [FaP97] worked on binary finite fields of the "tower" form $GF(2^{nm})$ where $m = 2^k$. The simplest case for the extension field of degree $m = 2$ was discussed in Kasahara & Morii [KM89] and Afanasyev [A91].

b. Using look-up tables of pre-computations

Guajardo [G97] and Guajardo & Paar [GP97] proposed a method that analyzes the complexity of the application of the Karatsuba-Ofman Algorithm (KOA) (discussed in [KO63]) and introduces a fast multiplication method in composite extension Galois fields of the form $GF(2^{nm})$ by using look-up logarithm and anti-logarithm tables in the subfield $GF(2^n)$.

c. Implementations over composite extension finite fields

Implemantions of elliptic curves over composite finite fields are presented in many works by: Paar ([P93],[P96],[P99]), Bosselaers, de Gersem, Vandenberghe, Vandewalle & de Win [BGVVW96], Paar & Soria-Rodriguez [PS97], Guajardo [G97], Guajardo & Paar [GP97], Fleischmann, Paar & Roelse [FPR98], Fleischmann, Paar & Soria-Rodriguez [FPS99] and Bailey, Paar & Woodbury [BPW00].

Many research articles have already focused on VLSI architectures for fast implementations of arithmetic operations: multiplication, inversion and exponentiation. Current approaches are combinations of structure of composite extension finite fields and

hardware architectures: bit parallel arithmetic in subfield and serial processing for extension field arithmetic. This is called the parallel-serial (hybrid) approach. This could have very fast implementations. Refer to Paar & Soria-Rodriguez [PS97] and its references.

e. Current issues on elliptic curves over composite extension finite fields

Consider a non-supersingular elliptic curve over a composite extension finite field $GF(2^{nm})$ of the equation: E: $y^2 + xy = x^3 + ax^2 + b$, where coefficients a and b are in the subfield $GF(2^n)$. Then E is referred to as a "subfield elliptic curve". Particularly, when $n = 1$, it is just a Koblitz curve or a binary anomalous curve (ABC).

In practice, we prefer the order $\#E$ to be prime or divisible by a small number. Hence we should use $GF(2^{nm})$ where n is equal to 1 or is small and m is a large prime. Otherwise, the order $\#E(GF(2^{nm}))$ will have a considerably large factor $\#E(GF(2^n))$.

However, there are always concerns about using the elliptic curves over composite extension Galois fields in cryptography by world mathematicians. Refer to Müller & Paulus [MP97].

There is also current research mentioning general doubts about constructing elliptic curves with special coefficients (such as Koblitz curves and subfield elliptic curves) and/or over finite fields with special internal structure (such as composite extension finite fields).

Gallant, Lambert & Vanstone [GLV00] showed that the parallelized Pollard lambda method can be improved by a factor of $(2m)^{1/2}$ for binary anomalous curves over finite fields $GF(2^m)$. The idea is to partition the group $\langle P \rangle$ into equivalence classes using the Frobenius endomorphisms Φ: $E(GF(2^m)) \rightarrow E(GF(2^m))$, by: $\Phi(x, y) = (x^2, y^2)$. We define the equivalence relation $\sim$ by: $P_1 \sim P_2$ if and only if $P_1 = \pm \Phi^l(P_2)$, for some l such that $0 \le l \le m - 1$. Assuming that $\Phi(P) = \Phi(x, y) = \lambda \cdot (x, y)$, then the equivalent class of point P includes $[P] = \{P, \lambda P, \lambda^2 P, \ldots, \lambda^{m-1} P, -P, -\lambda P, -\lambda^2 P, \ldots, -\lambda^{m-1} P\}$ and $[O] = \{O\}$. Therefore, the number of elements to be searched is reduced by a factor of $2m$; hence the running time that is proportional to the square root of the size of the group, will be reduced by a factor of $(2m)^{1/2}$.

Wiener & Zuccherato [WZ98] also showed the same improvement, not only for binary anomalous curves, but also for more generalized cases, subfield elliptic curves (defined over composite extension finite field $GF(2^m)$ with coefficients in the subfield $GF(2^n)$). The running time is also reduced by a factor of $(2m)^{1/2}$.

4.C.5. Representing points on an elliptic curve

The coordinates x and y of any point (x, y) on an elliptic curve must satisfy the cubic relation. Hence to represent an elliptic curve point, both coordinates are not required. Therefore, we can save space in storage of such points. There are a few methods developed to represent elliptic curve points. The terminology of such methods is not agreed upon globally yet.

a. Compressing and recovering points on an elliptic curve

When $p > 3$, E: $y^2 = x^3 + ax + b$ over a prime finite field F_p. The compressing and recovering employs the propert that: the coordinates of two points P and its (additive) inverse point $(-P)$ are: $P = (x, y)$ and $(-P) = (x, -y) = (x, p - y)$.

Compressing: Consider a point $P = (x_P, y_P)$ on E. Then the compressed form of P consists of x_P and the rightmost bit of y_P, denoted by $\tilde{y}_P$, when y_P is written in the binary

expansion form. In other words, we have $\tilde{y}_P \equiv y_P \pmod 2$. Two values y and $(p - y)$ always have the opposite rightmost bits.

Recovering: Given x_P and $\tilde{y}_P$, we can recover point P or, in fact, y_P. First, we compute the square root r of value $(x_P^3 + ax_P + b) \pmod p$. If the rightmost bit of r is equal to $\tilde{y}_P$, then $y_P = r$. Otherwise, let $y_P = p - r$.

When $p = 2$, E: $y^2 + xy = x^3 + ax^2 + b$, a non-supersingular elliptic curve over a finite finite field $GF(2^m)$. For any two points $P = (x, y)$ and $-P = (x, y + x)$, the difference between two ratios of point coordinates (yx^{-1}) and $(y + x)\, x^{-1} = yx^{-1} + 1$ are only 1, i.e., the rightmost bit in binary expansion form. Observe also that if $x \neq 0$, we can write the elliptic curve equation in terms of (yx^{-1}): $(yx^{-1})^2 + yx^{-1} = x + a + bx^{-2}$.

Compressing: The compressed form of a point $P = (x_P, y_P)$ on E consists of x_P and a bit $\tilde{y}_P$. If $x_P = 0$, let $\tilde{y}_P = 0$. (Actually, we do not care nor use this bit). If $x_P \neq 0$, let $\tilde{y}_P$ be the rightmost bit of $(y_P x_P^{-1})$.

Recovering: Given x_P and $\tilde{y}_P$, we can recover y_P as follows. If $x_P = 0$, let y_P be the square root of b. Particularly, using the identity $b^{2^m} = b$, we have $y_P = b^{2^{m-1}}$. (We ignore the bit $\tilde{y}_P$, or just consider it a check bit). If $x_P \neq 0$, we need to solve the equation $r^2 + r = x_P + a + bx_P^{-2} \pmod p$ for a root r_o. Observe that the other root is $(r_o + 1)$. We choose $r = r_o$ or $r = r_o + 1$, such that the rightmost bit of r is equal to $\tilde{y}_P$. Then compute $y_P = x_P r$.

b. Compact form for cyclic subgroup of an elliptic curve

This form, originally proposed by Seroussi [S98] is applied only for non-supersingular elliptic curves E: $y^2 + xy = x^3 + ax^2 + b$, with $b \neq 0$, over a binary finite field $GF(2^m)$. In fact, it is applied only for a cyclic subgroup of the elliptic curve. We can rewrite the equation as:

$$z^2 + z = x + a + bx^{-2} \text{ where } z = y/x, \text{ assuming that } x \neq 0.$$

We have some observations. Given $x \neq 0$, the above equation of z has a solution if and only if we have $Tr\,(x + a + bx^{-2}) = 0$. Therefore, for any point $P = (x, y) \in E$, where its x-coordinate $\neq 0$, we must have the identity: $Tr(x + a + bx^{-2}) = 0$. Using the equality: $Tr(a + b) = Tr(a) + Tr(b)$, it can be rewritten as: $Tr(x + bx^{-2)} + Tr(a) = 0$ or $Tr(x + bx^{-2)} = Tr(a)$, since values of the trace function $Tr(\cdot)$ is in F_2. The point $P = (0, y)$ in fact has order 2.

If $Q = (x_Q, y_Q) = 2P \in E$, then $x_Q = x^2 + bx^{-2}$. If $P \neq (0, y)$, then Q is not the point at infinity. Using the equality on trace function: $Tr(x) = Tr(x^2)$ over finite field $GF(2^m)$, we can derive the following relations: $Tr(x_Q) = Tr(x^2 + bx^{-2)} = Tr(x + bx^{-2)} = Tr(a)$, for any point P whose order is other than 2.

Seroussi used this fact to represent a point on an elliptic curve by "compact form" that needed only m bits, instead of $(m + 1)$ bits as they did in the compressed form, discussed above. In an elliptic curve cryptosystem, we should always consider points in a cyclic subgroup of large prime order n (hence n is odd). Any point $P = (x, y)$ in such subgroup cannot have order 2, and there always exists a point R such that $P = 2R$. Indeed, we can write explicitly: $P = (n + 1) \cdot P = 2R$, where $R = [(n + 1)/2] \cdot P$.

In other words, we always have $Tr(x_P) = Tr\,(a)$, for any point $P = (x, y) \neq (0, y)$ that is used in an elliptic curve cryptosystem. That is, we can eliminate one bit from the x-coordinate of point $P = (x, y)$ without ambiguity. The position of this bit can be chosen depending on the basis of the finite fields.

Combined with the compressed form, we can fill up the removed bit from x by the single bit representing y-coordinate. Hence this form has exactly m bits to represent a point $P = (x, y)$ in the cyclic subgroup mentioned.

Particularly, we will utilize the matrix implementation of trace function $Tr(x) = T.x^t$, where x^t is the transpose matrix of $x = (x_0, x_1, \ldots, x_{m-1}) \in GF(2^m)$ and T is an $m \times m$ matrix that depends on the basis used to represent the finite field $GF(2^m)$ and is calculated in advance. Then we will eliminate any i-th position bit of x where the corresponding bit $T(i) = 1$ in the matrix T. For recovering, that bit will be determined uniquely to get the relation between x and a: $Tr(x) = Tr(a)$.

This representation is proved to be optimal for points on non-supersingular elliptic curves over finite field $GF(2^m)$. Recall that a non-supersingular elliptic curve has even order and more particularly, $\#E(GF(2^m)) \equiv 2Tr(a) \pmod 4$. That is, one can write $\#E(GF(2^m)) = 2s$, for some value s (or even $\#E(GF(2^m)) = 4s$, for the case $Tr(a) = 0$). Therefore, the prime order n of the interested cyclic subgroup of the elliptic curve is at most ½ (or ¼ when $Tr(a) = 0$) of the curve order $\#E(GF(2^m))$ that is at most $(2^m + 1 + 2.2^{m/2})$. We can even drop the term 1 since the order must be even. Hence

If $Tr(a) = 1$, then $n \le 2^{m-1} + 2^{m/2} < 2^m$, for $m \ge 3$. Hence, an m-bit form can be sufficient to represent all points in the cyclic subgroup of the elliptic curve.

If $Tr(a) = 0$, then $n \le 2^{m-2} + 2^{(m/2)-1} < 2^{m-1}$, for $m \ge 3$. Hence, an $(m - 1)$-bit form is sufficient.

c. Other discussions

The compressing techniques just solve the simple problem of only 1-bit ambiguity of the y-coordinate. When we use only the x-coordinate, it does not matter how one can determine its corresponding y-coordinate of point P or its inverse point $(-P)$. If we need the y-coordinate, we still can use other conventions without having explicitly the extra bit, from the known facts on the coordinates of point P and point $(-P)$ as follows:

$y_{(-P)} = p - y_P$ for prime finite fields F_p, $p > 3$, and

$y_{(-P)} = x_P + y_P$ for binary finite fields $GF(2^m)$.

One can define y to be the smaller/larger, odd/even or "positive/negative" (in sense of modulo p) of two values upon mutual agreement.

Another approach is to try to use values independent of points (or y-coordinates) such as y^2 or $y \cdot (x + y)$ in our algorithms. But this approach could cost us more than 1 bit. For discussions and algorithms in this approach, refer to Montgomery [M97], Demytko [D94] and Schroeppel [Sc00].

Another option that could be employed is to use the full y-coordinate of a point and two bits to represent x-coordinate, since for a given y, there are possibly three values of x from the elliptic curve equation.

However, the security of an elliptic curve cryptosystem does not depend on the representation of a point in either a compressed, non-compressed or compact form.

4.C.6. Half-point algorithms

It is reasonable and practically necessary to find algorithms to compute the half of a point, i.e., $(½ \cdot P)$ of a given point P on an elliptic curve. In other words, we need to solve the following problem:

Given a point P, find another point Q on the same elliptic curve such that $2Q = P$.

We will discuss the solutions of this problem for finite fields.

For prime finite field F_p, $p > 3$, E: $y^2 = x^3 + ax + b$. Let $Q = (x_Q, y_Q)$ and $2Q = P = (x_P, y_P)$. From the point doubling formula, we have $x_P = [(3x_Q^2 + a)/2y_Q]^2 - 2x_Q$.

Then x_Q is a root of the equation $0 = f(X) = X^4 - 4X^3x_P - 2X^2a - X(4ax_P + 8b) + a^2 - 4bx_P$. For each x_Q, we find the corresponding values of y_Q if they exist. We can verify directly that $2 \cdot Q = P$. In fact, the algorithm for solving the half-point problem is expected to have polynomial running time. One may need to apply some algorithm to compute the square root modulo prime p of that running time.

Over binary finite field $GF(2^m)$, we consider a non-supersingular elliptic curve E: $y^2 + xy = x^3 + ax^2 + b$. We have $x_P = x_Q^2 + bx_Q^{-2}$. Similarly, we solve the following equation for x_Q: $0 = f(X) = X^4 + X^2 x_P + b = Y^2 + Yx_P + b = 0$. This equation of variable Y has a root if and only if $Tr(x_P^{-2} b) = 0$. If so, then we have $x_Q = Y^{1/2} = Y^{2^{m-1}}$.

Recall that if P is in a cyclic subgroup of prime order n of the elliptic curve, the half-point always exists. Since order n of P is odd, one can write: $P = (n + 1) \cdot P = 2 \cdot Q$, where $Q = [(n+1)/2] \cdot P$.

For completeness, we may consider the case of a supersingular elliptic curve over binary finite field $GF(2^m)$, E: $y^2 + cy = x^3 + ax + b$. We can write $x_P = (x_Q^4 + a^2)c^{-2}$. Hence $x_Q = (x_P c^2 + a^2)^{1/4}$, that always exists in the finite field $GF(2^m)$.

This existence of the half-point is used in Seroussi [S98] to represent a point on a non-supersingular elliptic curve over a finite field $GF(2^m)$ by a compact form of only m bits.

Meyer & Müller [MM96] also discussed the half-point problem on an elliptic curve over a ring Z_N for composite number $N = pq$. The authors referred to the problem by a different name, square root of a point. More references on this problem are in [KMOV92], Demytko [D94], Boyd & Smith [BS95] and [Kn99]. They showed that solving the half-point algorithm enables solving the integer-factoring problem (IFP).

4.C.7. Modular multiplication algorithm

Modular multiplication techniques are also helpful to increase the speed of general computation process. Montgomery's modular multiplication algorithm is the most popular algorithm and were discussed widely in crypto literature: Montgomery [M85], Acar, Kaliski & Koç [AKK96] and Acar & Koç [AK98].

There are still many works on implementation and performance of elliptic curve cryptosystems. Readers may refer to: Agnew, Mullin, Onyszchuk & Vanstone [AMOV91], Mister, Preneel, Wiener & de Win [MPWW98], Hankerson, Hernandez & Menezes [HHM00] and Brown, Hankerson, Hernandez & Menezes [BHHM01] and others.

Appendices

Appendix A. Trace functions

1. Trace of a finite field element

Trace of an element a is a linear mapping Tr: $GF(p^m) \rightarrow F_p$ defined by

$$Tr(a) = \sum_{i=0}^{m-1} a^{p^i} = a + a^{p^1} + a^{p^2} + \cdots + a^{p^{m-1}}$$

Basic properties ✍

$Tr(a^p) = Tr(a)$.

$Tr(a + b) = Tr(a) + Tr(b)$, and generally,

$Tr(v \cdot a) = v.Tr(a)$, for $a, b \in GF(p^m)$, $v \in F_p$.

When $p = 2$, $Tr(0) = 0$, and $Tr(1) = 1$ if m is odd and $Tr(1) = 0$ if m is even.

These above properties can be checked easily from the definition and the equality formula for finite fields of characteristic p: $(a + b)^p = a^p + b^p$, for all a, b in F_p.

Property: *For an element* $a \in GF(2^m)$, $Tr(a)$ *equals* 0 *for one half of the elements in* $GF(2^m)$ *and equals* 1 *for the other half.*

PROOF: When m is odd, we have $Tr(a + 1) = Tr(a) + Tr(1) = Tr(a) + 1$. Hence the mapping $z \mapsto (z + 1)$ is a bijection between the subset of elements of trace 0 and that of elements of trace 1. When m is even, the result still holds. However, the bijection in this case is $z \mapsto (z + b)$, where b is some element with $Tr(b) = 1$. (Such element b always exists.) q.e.d.

When $p > 2$, there are p^{m-1} elements of trace 0 in $GF(p^m)$. In fact, there is also equal distribution of values of trace function $Tr(\cdot)$ in the finite field $GF(p^m)$.

Property: *For an element* $a \in GF(2^m)$, *its trace* $Tr(a) = 0$ *if the polynomial* $(X^2 + X + a)$ *is reducible over* $GF(2^m)$ *or, in other words, has two roots over* $GF(2^m)$. *Conversely,* $Tr(a) = 1$ *if the polynomial* $(X^2 + X + a)$ *is irreducible over* $GF(2^m)$ *or it has no root over* $GF(2^m)$.

PROOF: Consider the homomorphism f: $GF(2^m) \to GF(2^m)$, defined by $f(X) = X^2 + X$. Its kernel is F_2, (since $f(0) = f(1) = 0$) and its image $\mathrm{Im}(f)$ is a subgroup of index 2 in $GF(2^m)$. Therefore, the polynomial $(X^2 + X + a)$ has a root (or it is reducible) over finite field $GF(2^m)$ if and only if we have $a = -a \in \mathrm{Im}(f)$. For such element a, we write: $a = b + b^2$, for some element $b \in GF(2^m)$. Hence $Tr(a) = Tr(b + b^2) = 0$.✍ q.e.d.

2. Trace of an elliptic curve

Recall that order of the group $E(F_q)$ is $\#E(F_q) = q + 1 - t$. We call t the trace of the elliptic curve E. Recall the definition of Frobenius endomorphism $\Psi \in End(E)$ of an

---

✍ **Number theory tip – The n^{th} roots of 1**

The multiplicative subgroup $GF(p^m)^*$ is a cyclic group of order $(p^m - 1)$, generated by a primitive element g. Let n be a divisor of $(p^m - 1)$. There are n roots of the polynomial $(x^n - 1)$. That are also called the n^{th} roots of 1, are n elements of the form $g^{j.(p^m-1)/n}$, for $0 \le j \le n - 1$. Particularly, for $n = p - 1$, we have:

$g^a \in GF(p)^*$ if and only if a is a multiple of $(p^m - 1)/(p - 1)$.

✍ **Number theory tip – Using the trace of a finite field element**

The equation $X^2 + aX + b = 0$, where $a, b \in GF(2^m)$, $a \neq 0$, has a root in $GF(2^m)$ if and only if $Tr(a^{-2}b) = 0$. Moreover, if x is one root of the equation, then $(x + a)$ is the other root. In other words, the number of solutions of the equation $X^2 + aX + b = 0$ equals $[2 - 2.Tr(a^{-2}b)]$ or $[1 + (-1)^{Tr(a^{-2}b)}]$. When $a = 0$, there is one root $x = (-b)^{1/2} = b^{2^{m-1}}$.

elliptic curve E by: $\forall(x, y) \in E$, $\Psi(x, y) = (x^q, y^q)$ and $\Psi(O) = O$. Then the trace t of the elliptic curve E also satisfies the relation $\Psi^2 - t\Psi + q = 0$ that is called the characteristic equation.

When $t = 0$, $\#E(F_q) = q + 1$, the elliptic curve is called supersingular.

When $t = 1$, $\#E(F_q) = q$, the elliptic curve is called F_p-anomalous. We will discuss later that these two types of elliptic curves are cryptographically insecure.

The elliptic curve with $t = 2$, or its order $\#E(F_q) = q - 1$, should also deserve some attention.

3. Properties on order of an elliptic curves

Property 1 (Lay & Zimmer [LZ94]): *Let E be a non-supersingular elliptic curve E*: $y^2 + xy = x^3 + ax^2 + b$, *over the finite field GF*(2^m). *Then the order of the elliptic curve E is* $\#E \equiv 2Tr(a) \pmod 4$.

PROOF: (Koblitz, using division polynomials). Let $Y = y/x$. We re-write the elliptic curve equation as: $Y^2 + Y = x + a + bx^{-2}$. Consider the division polynomial (that will be described later): $f_4(x) = x^6 + bx^2 = x^2(x^4 + b)$. The non-trivial points $Q = (x, y) \in E$ of order 4 will have $x = b^{1/4} = b^{2^{m-2}}$. The equation will have solution in $GF(2^m)$ (i.e., corresponding to the y-coordinates of Q) if and only if $0 = Tr(x + a + bx^{-2)} = Tr(b^{2^{m-2}} + a + b^{2^{m-1}}) = Tr(a)$,

since we have $Tr(B + B^2) = 0$ for any B. (Here $B = b^{2^{m-2}}$ and $B^2 = b^{2^{m-1}}$.) Thus the elliptic curve has a non-trivial point of order 4 if and only if $Tr(a) = 0$.

Since the order of a non-supersingular elliptic curve is even, or $\#E \equiv 0$ or 2 (mod 4), we can obtain the modular identity: $\#E \equiv 2.Tr(a) \bmod 4$. q.e.d.

This result shows that the maximum prime order of a cyclic subgroup of any non-supersingular elliptic curve is equal to ½ or ¼ the order of the curve itself. It is this cyclic subgroup that has interesting cryptographic uses in elliptic curve cryptosystems.

Property 2 (Seroussi [S98]): *Let E be a non-supersingular elliptic curve E*: $y^2 + xy = x^3 + ax^2 + b$ *over the finite field GF*(2^m). *Then for any point P of order other than* 2 *on E, the point* $Q = 2.P = (x_Q, y_Q)$ *will satisfy the condition:* $Tr(x_Q) = Tr(a)$.

This property later gives a way to represent a point of an elliptic curve over a finite field $GF(2^m)$ using only m bits.

Property 3 (J. Silverman): *The elliptic curve E*: $y^2 = x^3 + x$ *over a prime finite field* F_p *has its order satisfying the modular condition*: $\#E(F_p) \equiv 0 \pmod 4$.

PROOF: When $p \equiv 1 \pmod 4$, element (-1) is a quadratic residue in F_p. Hence, for each value of x such that the equation $y^2 = x^3 + x$ has two non-zero solutions of y, then the equation $y^2 = (-x)^3 + (-x) = -(x^3 + x)$ also has two non-zero solutions. Thus, we counted already a multiple of four points together. The rest in the set of points are the point at infinity O and 3 points whose y-coordinate is 0: (0, 0), $(t, 0)$, and $(-t, 0)$, where $t^2 = -1 \pmod p$. This modular equation always has two solutions, since the Legendre symbol modulo p of (-1), where $p \equiv 1 \pmod 4$ is 1. In summary, the number of elliptic curve points is a multiple of 4.

When $p \equiv 3 \pmod 4$, we have $\#E(F_p) = p + 1 \equiv 0 \pmod 4$. q.e.d.

Appendix B. Twisted curves

When char(K) = p > 3, two elliptic curves E: $y^2 = x^3 + ax + b$ and E': $y^2 = x^3 + ac^2x + c^3b$ where c is a quadratic non-residue (mod p), are twisted curves over the prime finite field F_p.

When char(K) = 2, two non-supersingular elliptic curves over the binary finite field $GF(2^m)$: E: $y^2 + xy = x^3 + (a + d)\,x^2 + b$ and E': $y^2 + xy = x^3 + ax^2 + b$, where d is any element of trace 1, are twisted curves.

When the finite field $GF(2^m)$ has the odd degree m over finite field F_2, we can choose $d = 1$. More generally, we will observe that:

For an element $d \in GF(2^m)$, its trace $Tr(d) = 1$ if and only if the polynomial $(X^2 + X + d)$ is irreducible over the finite field $GF(2^m)$.

Twisted curves have the same j-invariant. Their orders are related by: $\#E(F_q) = q + 1 - t$ and $\#E'(F_q) = q + 1 + t$. Twisted curves will be used in many applications, such as Demytko's elliptic curve cryptosystem and one-way permutations…

Theorem: *Let E be an ordinary elliptic curve over the finite field F_p and E' its twisted curve. Then we have:*

(i) $\#E(F_q) + \#E'(F_q) = 2q + 2$,

(ii) $E(GF(q^2)) \approx E'(GF(q^2))$. *That is, two elliptic curves are isomorphic over the quadratic extension field $GF(p^2)$ but not over the finite field F_q.*

We show the proof of statement (i) here since it is elementary, interesting and helpful. The proof for statement (ii) can be found in other elliptic curve literature, since it involves more difficult details.

PROOF: Over prime finite field F_p, (Kaliski [K91]):

Since the element c is a quadratic non-residue (mod p), then either $(x^3 + ax + b)$ or $c^3(x^3 + ax + b)$ is a quadratic residue modulo q, not both. Hence each pair of elements x and $cx \in F_p$ describes exactly a pair of points on elliptic curves $E(F_p)$ and $E'(F_p)$. We summarize the results in the table A.

Status of $(x^3 + ax + b)$	A pair of points on elliptic curves E and E'
$(x^3 + ax + b)$ is a quadratic residue	$(x, \pm(x^3 + ax + b)^{1/2})$ on E
$(x^3 + ax + b)$ is a quadratic non-residue	$(cx, \pm[c^3(x^3 + ax + b)]^{1/2})$ on E'
$(x^3 + ax + b) = 0$ (mod p)	$(x, 0)$ on E and $(cx, 0)$ on E'

Table A. The points on twisted curves $E(F_p)$ and $E'(F_p)$

These $2q$ points above, together with the two points at infinity O of two curves give us a total of $2(q + 1)$ points.

Over binary finite field $GF(2^m)$, (Meier & Staffelbach [MS93]):

We have $Tr(d) = 1$. For every fixed value $x \neq 0$, we will count the numbers of solutions for two equations in variable $Y = y/x$ corresponding to two curves: $Y^2 + Y + e_x = 0$ and $Y^2 + Y + (e_x + d) = 0$, for a constant $e_x = (x^3 + ax^2 + b)/x^2$. Because of $Tr(d) = 1$, there is only one elliptic curve equation that has two solutions and the other equation must have no solution. They accounted for $2\times(2^m - 1)$ points for both elliptic curves.

When $x = 0$, both elliptic curve equations are of the form $y^2 = b$. They always have one solution, $y = b^{1/2} = b^{2^{m-1}}$. These two points, together with two points at infinity O on two elliptic curves, complete our counting.

q.e.d.

Lemma: *Over prime finite field F_p two elliptic curves: $y^2 = x^3 + ax + b$ and $y^2 = x^3 + ac^2x + c^3b$, where c is a quadratic residue modulo p, have the same order.*

Over binary finite field $GF(2^m)$, two elliptic curves: $y^2 + xy = x^3 + ax^2 + b$ and $y^2 + xy = x^3 + (a + e)\,x^2 + b$, where $Tr(e) = 0$, have the same order. (Moreover, they are isomorphic elliptic curves.)

Over $GF(2^m)$, two elliptic curves:$y^2 + xy = x^3 + ax^2 + b$ and $y^2 + xy = x^3 + a^2x^2 + b$, have the same order. (Moreover, they are isomorphic elliptic curves.)

PROOF: The first two statements are straightforward from the above theorem. The third statement is a simple result of the second one where $e = a^2 + a$. Then we have: $Tr(e) = Tr(a^2) + Tr(a) = 0$. q.e.d.

Appendix C. Examples of elliptic curves over small binary finite fields $GF(2^m)$

Example 1

We consider the finite field $GF(2^2)$. There is only one irreducible polynomial $f(x) = x^2 + x + 1$ over F_2. Let $\alpha = x \pmod{f(x)}$ be a root of $f(x)$. The four elements of $GF(2^2)$ are 0, 1, α and $\alpha^2 = \alpha + 1$. Obviously, $\alpha^3 = 1$, or we say the element α is the third root of unity in the field. In fact, α and α^2 are two primitive elements of the multiplicative subgroup $GF(2^2)^*$ and are the only roots of $f(x)$. Hence $f(x)$ is the only primitive polynomial of $GF(2^2)$.

The trinomial basis is $\{1, \alpha\}$ and $\{\alpha, \alpha^2\}$ is an ONB (of both Type I and II.) For example, we can write: $1 = \alpha^2 + \alpha$. Hence, in trinomial basis, we have: $GF(2^2) = \{(00) = 0, (01) = 1, (10) = \alpha, (11) = \alpha^2\}$. And in ONB, we write: $GF(2^2) = \{(00) = 0, (11) = 1, (10) = \alpha, (01) = \alpha^2\}$. Observe that $Tr(0) = Tr(1) = 0$ and $Tr(\alpha) = Tr(\alpha^2) = \alpha + \alpha^2 = 1$. Hence the trinomial basis is not self-dual. The ONB is self-dual and is also a primitive normal basis. Now we define $g(z) = Tr(\alpha^2 z)$, $\forall z \in GF(2^2)$. Then $\{1, \alpha\}$ is its dual basis with respect to the linear function $g(\cdot)$.

Consider the elliptic curve of the equation E: $y^2 + xy = x^3 + x^2 + \alpha$. Its order is $\#E = 4$ and its elements are $\{O, (0,\alpha^2), (\alpha,0), (\alpha,\alpha)\}$. We can compute simple point multiplications: $2(\alpha,0) = (0,\alpha^2)$. Hence $4(\alpha,0) = O$. Then point $(\alpha, 0)$ is a generator of E. The other generator is, obviously, the point (α,α). We can check that: $2(\alpha,\alpha) = (0,\alpha^2)$.

Let E: $y^2 + xy = x^3 + x^2 + 1$ be a Koblitz curve. The group E has order 8 and its structure is $Z_{n_1} \oplus Z_{n_2}$, where $n_2 = \gcd(n_1, q - 1)$. Since n_1 is a divisor of 8 and $q - 1 = 7$; hence $n_2 = 1$ and $E = Z_8$, a cyclic group. We have

$$E = \langle(\alpha, 1)\rangle = \{(0,1), (1,\alpha), (1,\alpha^2), (\alpha, 1), (\alpha,\alpha^2), (\alpha^2,1), (\alpha^2,\alpha), O\}.$$

The other generators are points (α, α^2), $(\alpha^2,1)$ and (α^2, α).

Example 2 (Koblitz, Menezes & Vanstone [KMV96])

The finite field $GF(2^3)$ is a vector space of dimension 3 over F_2. Its irreducible polynomial is chosen to be $f(x) = x^3 + x + 1$ whose root is denoted by $\alpha = x \pmod{f(x)}$. We list all elements of the multiplicative subgroup $GF(2^3)^*$ as powers of α. Obviously, we have $\alpha^7 = 1$. In fact, all six non-trivial powers of α are primitive elements of the multiplicative subgroup $GF(2^3)^*$.

$\alpha^0 = (001) = 1$	$\alpha^1 = (010)$	$\alpha^2 = (100)$	$\alpha^3 = (011) = \alpha + 1$	$\alpha^4 = (110) = \alpha^2 + \alpha$	$\alpha^5 = (111) = \alpha^2 + \alpha + 1$	$\alpha^6 = (101) = \alpha^2 + 1$

Table C.1. Elements in the finite field $GF(2^3)$ using trinomial basis $\{1, \alpha, \alpha^2\}$

The polynomial $f(x) = x^3 + x + 1$ is primitive and its roots are: α, α^2 and α^4. The only other primitive polynomial for $GF(2^3)$ is $g(x) = (x + \alpha^3)(x + \alpha^5)(x + \alpha^6) = x^3 + x^2 +$

1. Their roots included all six primitive elements in $GF(2^3)^*$. Note that we still have the same results if we do computations over other bases, e.g., ONB. These two polynomials are also the only irreducible polynomials of $GF(2^3)$.

We can compute $Tr(1) = 1$, $Tr(\alpha^3) = Tr(\alpha + 1) = 1$ and $Tr(\alpha) = \alpha + \alpha^2 + \alpha^4 = 0 = Tr(\alpha^2) = Tr(\alpha^4)$. Hence the trinomial basis $\{1, \alpha, \alpha^2\}$ is not self-dual with respect to $Tr(\cdot)$. But the permutation $\{1, \alpha^2, \alpha\}$ is its dual basis.

Consider the elliptic curve E: $y^2 + xy = x^3 + x^2 + 1$ over the finite field $GF(2^3)$. Refer to Figure 1.3. We will check that $\#E = 14$. In fact, E is a Koblitz curve over F_2, and has only two points, (0,1) and O. Using Hasse's theorem, we can write $\#E(F_2) = 2 + 1 - t = 2$, since $t = 1$. The equality: $2T^2 - T + 1 = (1 - aT)(1 - bT)$ gives us two relations: $a + b = 1$ and $ab = 2$. Then we have: $a^2 + b^2 = (a + b)^2 - 2ab = -3$ and $a^3 + b^3 = (a + b)[(a + b)^2 - 3ab)] = -5$. Hence $\#E = 2^3 + 1 - (a^3 + b^3) = 14$. The order is square-free; hence E is a cyclic group. It is generated by point $P = (\alpha, \alpha^5)$. Note that, in this example as well as other examples in this document, we always use the explicit point addition rules given in the table 1.2. Check that

$$7P = 2(2P + P) + P = 2[(\alpha^3, 0) + (\alpha, \alpha^5)] + (\alpha, \alpha^5)$$
$$= 2(\alpha^2, \alpha^5) + (\alpha, \alpha^5) = (\alpha^6, \alpha^6) + (\alpha, \alpha^5) = (0,1).$$

Hence $14P = O$. More explicitly, we have

$$E = \langle(\alpha, \alpha^5)\rangle = \{P = (\alpha, \alpha^5), 2P = (\alpha^3, 0), 3P = (\alpha^2, \alpha^5), 4P = (\alpha^4, 0), 5P = (\alpha^4, \alpha^3),$$
$$6P = (\alpha^6, \alpha^6), \mathbf{7P =} (0,1), 8P = (\alpha^6, 0), 9P = (\alpha^4, \alpha^6), 10P = (\alpha^5, \alpha^5),$$
$$11P = (\alpha^2, \alpha^3), 12P = (\alpha^3, \alpha^3), 13P = (\alpha, \alpha^6), O\}.$$

In fact, the other five generators for E are points $3P$, $5P$, $9P$, $11P$ and $13P$.

Example 3

The finite field $GF(2^3)$ can be represented in the optimal normal basis of type II. It is the only normal basis of $GF(2^3)$. Its irreducible polynomial is $f(x) = x^3 + x^2 + 1$ whose root is denoted by $\alpha = x \pmod{f(x)}$. This polynomial is a primitive polynomial as we discussed in the previous example 2. In fact, all six non-trivial powers of α are primitive elements of the multiplicative subgroup $GF(2^3)^*$. They are six non-trivial 7^{th} roots of unity. We can list all elements of the cyclic group $GF(2^3)^*$ using Type II ONB $\{\alpha, \alpha^2, \alpha^4\}$ in the table C.2.

$\alpha^0 = (111) = \alpha + \alpha^2 + \alpha^4$	$\alpha^1 = (100)$	$\alpha^2 = (010)$	$\alpha^3 = (101) = \alpha + \alpha^4 = \alpha^2 + 1$	$\alpha^4 = (001) = \alpha^2 + \alpha + 1$	$\alpha^5 = (011) = \alpha^2 + \alpha^4 = \alpha + 1$	$\alpha^6 = (110) = \alpha + \alpha^2$

Table C.2. Elements in the finite field $GF(2^3)$ using Type II ONB $\{\alpha, \alpha^2, \alpha^4\}$

Recall the trace formula $Tr(a) = Tr(a_0, a_1, \ldots, a_{m-1}) = a_0 \oplus a_1 \oplus \ldots \oplus a_{m-1}$, we get:

$Tr(1) = Tr(\alpha) = Tr(\alpha^2) = Tr(\alpha^4) = 1$ and $Tr(0) = Tr(\alpha^3) = Tr(\alpha^5) = Tr(\alpha^6) = 0$.

Hence the above ONB is also self-dual. In fact, it is the only self-dual normal basis here. Furthermore, it is a primitive normal basis since α is primitive in the subgroup $GF(2^3)^*$.

The elliptic curve E: $y^2 + xy = x^3 + x^2 + 1$ over the finite field $GF(2^3)$ has order 14 and its elements are generated by $R = (\alpha^3, \alpha)$. Check that

$$7R = 2(2R + R) + R = 2[(\alpha^2, 0) + (\alpha^3, \alpha)] + (\alpha^3, \alpha)$$
$$= 2(\alpha^6, \alpha) + (\alpha^3, \alpha) = (\alpha^4, \alpha^4) + (\alpha^3, \alpha) = (0,1).$$

Hence $14R = O$. More explicitly, we have

$E = \langle(\alpha^3, \alpha)\rangle = \{R = (\alpha^3, \alpha), 2R = (\alpha^2, 0), 3R = (\alpha^6, \alpha), 4R = (\alpha, 0), R = (\alpha^5, \alpha^2),$
$6R = (\alpha^4, \alpha^4), 7R = (0, 1), 8R = (\alpha^4, 0), 9R = (\alpha^5, \alpha^4), 10R = (\alpha, \alpha),$
$11R = (\alpha^6, \alpha^2), 12R = (\alpha^2, \alpha^2), 13R = (\alpha^3, \alpha^4), \mathrm{O}\}$.

Any of the six points, R, $3R$, $5R$, $9R$, $11R$ and $13R$, can be a generator for E.

The twisted curve of E is $\tilde{E}$: $y^2 + xy = x^3 + 1$. It has order $\#\tilde{E}(GF(2^3)) = 2(2^3 + 1) - 14 = 4$. We also have $\#\tilde{E}(F_2) = 2(2 + 1) - 2 = 4$. Hence: $\tilde{E}(F_2) = \tilde{E}(GF(2^3)) = \{\mathrm{O}, (0,1), (1,0), (1,1)\}$, and is generated by either point (1, 0) or (1, 1).

Example 4

We consider the finite field $GF(2^4)$. The trinomial $f(x) = x^4 + x + 1$ is irreducible over F_2. Then the non-zero elements of the cyclic subgroup $GF(2^4)^*$ can be generated by an element, $\alpha = x \pmod{f(x)}$, a root of function $f(x)$ in $GF(2^4)$. Obviously, $\alpha^{15} = 1$.

$\alpha^0 = (0001)$	$\alpha^1 = (0010)$	$\alpha^2 = (0100)$	$\alpha^3 = (1000)$	$\alpha^4 = (0011)$ $= \alpha + 1$
$\alpha^5 = (0110)$ $= \alpha^2 + \alpha$	$\alpha^6 = (1100)$ $= \alpha^3 + \alpha^2$	$\alpha^7 = (1011)$ $= \alpha^3 + \alpha + 1$	$\alpha^8 = (0101)$ $= \alpha^2 + 1$	$\alpha^9 = (1010)$ $= \alpha^3 + \alpha$
$\alpha^{10} = (0111)$ $= \alpha^2 + \alpha + 1$	$\alpha^{11} = (1110)$ $= \alpha^3 + \alpha^2 + \alpha$	$\alpha^{12} = (1111)$ $= \alpha^3 + \alpha^2 + \alpha + 1$	$\alpha^{13} = (1101)$ $= \alpha^3 + \alpha^2 + 1$	$\alpha^{14} = (1001)$ $= \alpha^3 + 1$

Table C.3. Elements in the finite field $GF(2^4)$ using trinomial basis $\{1, \alpha, \alpha^2, \alpha^3\}$

In fact, there are seven other elements, which can serve as a generator for the multiplicative subgroup $GF(2^4)^*$. They are α^2, α^4, α^7, α^8, α^{11}, α^{13} and α^{14}. The polynomial $f(x) = x^4 + x + 1$ is primitive and its roots are: α, α^2, α^4 and α^8. The only other primitive polynomial for the finite field $GF(2^4)$ is

$$g(x) = (x + \alpha^7)\,[(x + \alpha^7)^{2}][x + (\alpha^7)^{2^2}\,][x + (\alpha^7)^{2^3}\,].$$
$$= (x + \alpha^7)\,(x + \alpha^{14})\,(x + \alpha^{13})\,(x + \alpha^{11}) = x^4 + x^3 + 1.$$

Their roots included all eight primitive elements in the multiplicative subgroup $GF(2^4)^*$

Examples of finite field arithmetic:

$(1101) + (1001) = (0100)$, i.e., $(x^3 + x^2 + 1) + (x^3 + 1) = x^2 \bmod f(x)$, and

$(1101)\cdot(1001) = (1111)$, since $(x^3 + x^2 + 1) \cdot (x^3 + 1) = x^3 + x^2 + x + 1 \bmod f(x)$.

We still can do multiplications by representing finite field elements as powers of element α. $(1101)\cdot(1001) = \alpha^{13}$. $\alpha^{14} = \alpha^{27} = \alpha^{12} = (1111)$.

We can compute the trace of elements. For example, $Tr(\alpha^3) = \alpha^3 + \alpha^{3\times2} + \alpha^{3\times2^2} + \alpha^{3\times2^3} = \alpha^3 + \alpha^6 + \alpha^{12} + \alpha^9 = 1$. Then we have $Tr(\alpha^i) = 1$, when i = 3, 6, 7, 9, 11, 12, 13 and 14, and $Tr(\alpha^i) = 0$, if otherwise.

We can verify that two self-dual bases for the finite field $GF(2^4)$ are $\{\alpha^3, \alpha^7, \alpha^{12}, \alpha^{13}\}$ and $\{\alpha^6, \alpha^9, \alpha^{11}, \alpha^{14}\}$. For example, $Tr(\alpha^{7\times2}) = 1$, $Tr(\alpha^7\alpha^3) = Tr(\alpha^{10}) = 0\ldots$ They are not normal bases. The above trinomial basis is not a dual basis since we have $Tr(1) = 0$. Now we define $g(z) = Tr(\alpha^{-1}z)$, $\forall z \in GF(2^4)$. Then the permutation $\{1, \alpha^3, \alpha^2, \alpha\}$ is its dual basis with respect to the linear function $g(\cdot)$.

Example 5

Let E: $y^2 + xy = x^3 + x^2 + 1$ be a Koblitz curve. The group $E(GF(2^4))$ has order 16, by using Hasse's theorem. The Frobenius equation $2T^2 - T + 1 = (1 - aT)(1 - bT)$ gives us two relations: $a + b = 1$ and $ab = 2$. We can compute two power sums of a and b:

$a^2 + b^2 = (a + b)^2 - 2ab = -3$ and $a^4 + b^4 = (a^2 + b^2)^2 - 2(ab)^2 = 9 - 8 = 1$.

Then $\#E = 2^4 + 1 - (a^4 + b^4) = 16$. Hence E is an anomalous binary curve over both finite fields F_2 and $GF(2^4)$. Recall that the structure of the group is $Z_{n_1} \oplus Z_{n_2}$, where $n_2 \mid \gcd(n_1, q-1)$. Since n_1 is some divisor of 16 and $(q - 1) = 15$; hence $n_2 = 1$ and $E = Z_{16}$, a cyclic subgroup. Therefore, we can observe that for any point $P = (x, y) \in E$, then point $(x^2, y^2) \in E$, and of course, $(x^{2^4}, y^{2^4}) = (x, y) \in E$. This helps to reduce the task of listing all points on the elliptic curve (using ONB Type I in the previous example).

$E = \langle(\alpha^3, \alpha)\rangle = \{(0,1), (1, \alpha^5), (1, \alpha^{10}), (\alpha^3, \alpha), (\alpha^3, \alpha^9), (\alpha^5, 1), (\alpha^5, \alpha^{10}), (\alpha^9, \alpha^8),$
$(\alpha^9, \alpha^{12}), (\alpha^{10},1), (\alpha^{10}, \alpha^5), (\alpha^{12}, \alpha^4), (\alpha^{12}, \alpha^6), (\alpha^6, \alpha^2), (\alpha^6, \alpha^3), \mathrm{O}\}$.

Again, the other seven generators for E are scalar point multiplications of the form $a\cdot(\alpha^3,\alpha)$, where a runs through all odd numbers between 3 and 15.

Example 6

We consider the finite field $GF(2^5)$ using a polynomial basis. The trinomial $f(x) = x^5 + x^2 + 1$ is irreducible over F_2. The middle term of interested trinomials here can be either x or x^2 only. The other trinomial $(x^5 + x + 1)$ is reducible. Indeed, $x^5 + x + 1 = (x^2 + x + 1)(x^3 + x^2 + 1)$. Then the non-zero elements of the cyclic (multiplicative) subgroup $GF(2^5)^*$ can be generated by the single element, $\alpha = x \pmod{f(x)}$, a root of $f(x)$ in $GF(2^5)$.

$0 = (00000)$	$\alpha = (00010)$	$\alpha^2 = (00100)$	$\alpha^3 = (01000)$	$\alpha^4 = (10000)$
$\alpha^5 = (00101)$ $= \alpha^2 + 1$	$\alpha^6 = (01010)$ $= \alpha^3 + \alpha$	$\alpha^7 = (10100)$ $= \alpha^4 + \alpha^2$	$\alpha^8 = (01101)$ $= \alpha^3 + \alpha^2 + 1$	$\alpha^9 = (11010)$ $= \alpha^4 + \alpha^3 + \alpha$
$\alpha^{10} = (10001)$ $= \alpha^4 + 1$	$\alpha^{11} = (00111)$ $= \alpha^2 + \alpha + 1$	$\alpha^{12} = (01110)$ $= \alpha^3 + \alpha^2 + \alpha$	$\alpha^{13} = (11100)$ $= \alpha^4 + \alpha^3 + \alpha^2$	$\alpha^{14} = (11101) =$ $\alpha^4 + \alpha^3 + \alpha^2 + 1$
$\alpha^{15} = (11111)$ $= \alpha^4 + \alpha^3 + \alpha^2$ $+\alpha + 1$	$\alpha^{15} = (11011)$ $= \alpha^4 + \alpha^3 + \alpha + 1$	$\alpha^{17} = (10011)$ $= \alpha^4 + \alpha + 1$	$\alpha^{18} = (00011)$ $= \alpha + 1$	$\alpha^{19} = (00110)$ $= \alpha^2 + \alpha$
$\alpha^{20} = (01100)$ $= \alpha^3 + \alpha^2$	$\alpha^{21} = (11000)$ $= \alpha^4 + \alpha^3$	$\alpha^{22} = (10101)$ $= \alpha^4 + \alpha^2 + 1$	$\alpha^{23} = (01111) =$ $\alpha^3 + \alpha^2 + \alpha + 1$	$\alpha^{24} = (11110) =$ $\alpha^4 + \alpha^3 + \alpha^2 + \alpha$
$\alpha^{25} = (11001)$ $= \alpha^4 + \alpha^3 + 1$	$\alpha^{26} = (10111) =$ $\alpha^4 + \alpha^2 + \alpha + 1$	$\alpha^{27} = (01011)$ $= \alpha^3 + \alpha + 1$	$\alpha^{28} = (10110)$ $= \alpha^4 + \alpha^2 + \alpha$	$\alpha^{29} = (01001)$ $= \alpha^3 + 1$
$\alpha^{30} = (10010) = \alpha^4 + \alpha$		$1 = (00001) = \alpha^{31}$		

Table C.4. Elements in the finite field $GF(2^5)$ using trinomial basis $\{1, \alpha, \alpha^2, \alpha^3, \alpha^4\}$

In fact, all 30 non-trivial powers of α are primitive elements of the multiplicative subgroup $GF(2^5)^*$. There are 6 primitive polynomials for the finite field $GF(2^5)$. They are also all the irreducible polynomials for $GF(2^5)$, since $2^5 - 1 = 31$ is Mersenne prime number.

$(x + \alpha)(x + \alpha^2)(x + \alpha^4)(x + \alpha^8)(x + \alpha^{16}) = x^5 + x^2 + 1,$
$(x + \alpha^3)(x + (\alpha^3)^2)(x + (\alpha^3)^4)(x + (\alpha^3)^8)(x + (\alpha^3)^{16}) = x^5 + x^4 + x^3 + x^2 + 1,$
$(x + \alpha^5)(x + (\alpha^5)^2)(x + (\alpha^5)^4)(x + (\alpha^5)^8)(x + (\alpha^5)^{16}) = x^5 + x^4 + x^2 + x + 1,$
$(x + \alpha^7)(x + (\alpha^7)^2)(x + (\alpha^7)^4)(x + (\alpha^7)^8)(x + (\alpha^7)^{16}) = x^5 + x^3 + x^2 + x + 1,$
$(x + \alpha^{11})(x + (\alpha^{11})^2)(x + (\alpha^{11})^4)(x + (\alpha^{11})^8)(x + (\alpha^{11})^{16}) = x^5 + x^4 + x^3 + x + 1,$
$(x + \alpha^{15})(x + (\alpha^{15})^2)(x + (\alpha^{15})^4)(x + (\alpha^{15})^8)(x + (\alpha^{15})^{16}) = x^5 + x^3 + 1.$

Their roots included all 30 primitive elements of the multiplicative subgroup $GF(2^5)^*$.

We can compute the trace of elements. For example,

$Tr(1) = 1$, $Tr(\alpha^3) = \alpha^3 + \alpha^{3.2} + \alpha^{3.2^2} + \alpha^{3.2^3} + \alpha^{3.2^4} = 1$ and

$Tr(\alpha) = \alpha + \alpha^2 + \alpha^{2^2} + \alpha^{2^3} + \alpha^{2^4} = 0 = Tr(\alpha^{2^n})$, for positive integers n.

Then $Tr(\alpha^i) = 1$, when $i = 0, 3, 5, 6, 9, 10, 11, 12, 13, 17, 18, 20, 21, 22, 24$ and 26,

$Tr(\alpha^i) = 0$, if otherwise.

The trinomial basis $\{1, \alpha, \alpha^2, \alpha^3, \alpha^4\}$ is not self-dual, since, e.g., $Tr(\alpha^2) = 0$. If we define $g(z) = Tr(\alpha^{25}z)$, $\forall z \in GF(2^5)$. Then its permutation $\{\alpha, 1, \alpha^4, \alpha^3, \alpha^2\}$ is its dual basis with respect to the linear function $g(\cdot)$.

Consider another non-supersingular elliptic curve of the form: $E: y^2 + xy = x^3 + \alpha$:

$E = \{O, (0,\alpha^{16}), (\alpha,\alpha^{13}), (\alpha,\alpha^{24}), (\alpha^2,\alpha^{13}), (\alpha^2,\alpha^{21}), (\alpha^3,\alpha^{24}), (\alpha^3,\alpha^{28}), (\alpha^5,\alpha^{17}), (\alpha^5,\alpha^{28}),$
$(\alpha^6,\alpha^7), (\alpha^6,\alpha^{24}), (\alpha^8,\alpha^{16}), (\alpha^8,\alpha^{28}), (\alpha^{10},1), (\alpha^{10},\alpha^4), (\alpha^{11},1), (\alpha^{11},\alpha^{19}), (\alpha^{13},\alpha^2),$
$(\alpha^{13},\alpha^{21}), (\alpha^{14},\alpha^7), (\alpha^{14},\alpha^{29}), (\alpha^{15},\alpha), (\alpha^{15},\alpha^{14}), (\alpha^{19},\alpha^9), (\alpha^{19},\alpha^{13}), (\alpha^{21},0), (\alpha^{21},\alpha^{21}),$
$(\alpha^{26},\alpha^7), (\alpha^{26},\alpha^{18}), (\alpha^{28},1), (\alpha^{28},\alpha^{26})\}$.

Then $\#E(GF(2^5)) = 32 = \#GF(2^5)$. The structure of the group is $Z_{n_1} \oplus Z_{n_2}$, where $n_2 | \gcd(n_1, 31)$. Since n_1 is a divisor of 32, and 31 is prime; hence $n_2 = 1$ and $E = Z_{32}$, a cyclic group. Observe that the elliptic curve order $\#E = 32 \equiv 0 \pmod 4$, since $Tr(0) = 0$.

Two Koblitz curves $E_a: y^2 + xy = x^3 + ax^2 + 1$, where $a = 0$ or 1, over finite field $GF(2^5)$ have the orders $\#E_1 = 22$ and $\#E_0 = 44$. They are just twisted elliptic curves.

Consider a non-supersingular elliptic curve $E_2: y^2 + xy = x^3 + \alpha^2x^2 + 1$ over the finite field $GF(2^5)$ with the trinomial basis above.

$E_2 = \{O, (0,1), (1,\alpha^6), (1,\alpha^{27}), (\alpha,\alpha^{27}), (\alpha,\alpha^{29}), (\alpha^2,\alpha^{15}), (\alpha^2,\alpha^{16}), (\alpha^4,\alpha^3), (\alpha^4,\alpha^{21}),$
$(\alpha^5,\alpha^{14}), (\alpha^5,\alpha^{21}), (\alpha^8,\alpha^{16}), (\alpha^8,\alpha^{28}), (\alpha^9,\alpha^7), (\alpha^9,\alpha^{12}), (\alpha^{10},\alpha^{21}), (\alpha^{10},\alpha^{29}),$
$(\alpha^{11},\alpha^{20}), (\alpha^{11},\alpha^{27}), (\alpha^{13},\alpha^{17}), (\alpha^{13},\alpha^{23}), (\alpha^{15},1), (\alpha^{15},\alpha^{24}), (\alpha^{16},1), (\alpha^{16},\alpha^9),$
$(\alpha^{18},\alpha^{10}), (\alpha^{18},\alpha^{30}), (\alpha^{20},\alpha^{24}), (\alpha^{20},\alpha^{30}), (\alpha^{21},\alpha^2), (\alpha^{21},\alpha^{13}), (\alpha^{22},\alpha^2), (\alpha^{22},\alpha^{10}),$
$(\alpha^{23},\alpha^4), (\alpha^{23},\alpha^{15}), (\alpha^{26},\alpha^{24}), (\alpha^{26},\alpha^{29}), (\alpha^{27},\alpha^{25}), (\alpha^{27},\alpha^{30}), (\alpha^{29},\alpha^2), (\alpha^{29},\alpha^8),$
$(\alpha^{30},\alpha^{10}), (\alpha^{30},\alpha^{18})\}$.

Its order is 44. The structure of the group is $Z_{n_1} \oplus Z_{n_2}$, where $n_2 | \gcd(n_1, 31)$. Since n_1 is a divisor of 44, and 31 is prime; hence $n_2 = 1$ and $E_2 = Z_{44}$, a cyclic group.

Example 7

We consider the finite field $GF(2^5)$ using a normal basis. The reduction polynomial is $f(x) = x^5 + x^4 + x^2 + x + 1$. By the way, it is a primitive polynomial. The optimal normal basis of Type II consists of five polynomials: x, x^2, x^4, x^8 and x^{16} (mod $f(x)$). A generator for non-zero elements of multiplicative subgroup $GF(2^5)^*$ is chosen to be $\beta = x \pmod{f(x)}$. Indeed, since $GF(2^5)^*$ is a cyclic group of prime order (31), any element other than 1 can be a generator for the whole group. We can write them explicitly in the table C.5., where

$$(a_0, a_1, a_2, a_3, a_4) = a_0x + a_1x^2 + a_2x^4 + a_3x^8 + a_4x^{16} \pmod{f(x)}.$$

For instance, $1 = (11111) = x + x^2 + x^4 + x^8 + x^{16}$. (All polynomials are of modulo $f(x)$.) We can compute the following intermediate terms and the next four repeated squares of each term (by simple right 1-cylic shifts). They are:

$x^5 \equiv x^4 + x^2 + x + 1 = x^8 + x^{16} = (00011)$, and $x^{10} = (10001)$, $x^{20} = (11000)$

$x^7 \equiv x^2 + 1 = x + x^4 + x^8 + x^{16} = (10111)$,

$x^3 \equiv x + x^8 = (10010)$, since $x^8 \equiv x^3 + x$,

$x^{11} \equiv x^6 + x^4 = (x + x^8)^2 + x^4 = x^2 + x^4 + x^{16} = (01101)$ and

$x^{15} \equiv (x + x^8)^4 x^3 = (x + x^4) x^3 = x + x^8 + x^{16} = (10011)$.

$0 = (00000)$	$\beta = (10000)$	$\beta^2 = (01000)$	$\beta^3 = (10010)$	$\beta^4 = (00100)$
$\beta^5 = (00011)$	$\beta^6 = (01001)$	$\beta^7 = (10111)$	$\beta^8 = (00010)$	$\beta^9 = (01100)$
$\beta^{10} = (10001)$	$\beta^{11} = (01101)$	$\beta^{12} = (10100)$	$\beta^{13} = (01011)$	$\beta^{14} = (11011)$
$\beta^{15} = (10011)$	$\beta^{16} = (00001)$	$\beta^{17} = (00101)$	$\beta^{18} = (00110)$	$\beta^{19} = (01111)$
$\beta^{20} = (11000)$	$\beta^{21} = (11010)$	$\beta^{22} = (10110)$	$\beta^{23} = (00111)$	$\beta^{24} = (01010)$
$\beta^{25} = (11110)$	$\beta^{26} = (10101)$	$\beta^{27} = (01110)$	$\beta^{28} = (11101)$	$\beta^{29} = (11100)$
$\beta^{30} = (11001)$	$\beta^{31} = 1 = (11111)$			

Table C.5. Elements in the finite field $GF(2^5)$ using Type II ONB, $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}\}$

Then the field multiplications will be very simple on powers of β. Recall also that, in normal basis representation, the trace of an element can be computed easily by the formula: $Tr(a) = Tr(a_0, a_1, \ldots, a_{m-1}) = a_0 \oplus a_1 \oplus \ldots \oplus a_{m-1}$. Then we have: $Tr(\beta^i) = 1$, when $i = 0, 1, 2, 4, 8, 11, 13, 15, 16, 21, 22, 23, 26, 27, 29$ and 30, while $Tr(\beta^i) = 0$, if otherwise. The ONB, therefore, is a self-dual basis, since the trace of all elements of the basis is 1. It is also a primitive normal basis.

We can use the algorithm described in chapter 4 to compute the product terms $\lambda_{i,j}$. First, we create 4 matrices:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad S = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \text{ and matrix } T = S.A^{-1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

There are nine terms $\lambda_{i,j}$ which are all 1: (By $\lambda_{i,j} = T_{j-i,-i}$, where indices are of modulo $m = 5$) $\lambda_{0,1} = T_{1,0}$, $\lambda_{1,0} = T_{4,4}$, $\lambda_{1,3} = T_{2,4}$, $\lambda_{2,3} = T_{1,3}$, $\lambda_{2,4} = T_{2,3}$, $\lambda_{3,1} = T_{3,2}$, $\lambda_{3,2} = T_{4,2}$, $\lambda_{4,2} = T_{3,1}$ and $\lambda_{4,4} = T_{0,1}$. Then $(a_0, a_1, a_2, a_3, a_4) \cdot (b_0, b_1, b_2, b_3, b_4) = (c_0, c_1, c_2, c_3, c_4)$ where

$$\begin{aligned}
c_0 &= a_0 b_1 \oplus a_1 (b_0 \oplus b_3) \oplus a_2 (b_3 \oplus b_4) \oplus a_3 (b_1 \oplus b_2) \oplus a_4 (b_2 \oplus b_4) \\
c_1 &= a_1 b_2 \oplus a_2 (b_1 \oplus b_4) \oplus a_3 (b_4 \oplus b_0) \oplus a_4 (b_2 \oplus b_3) \oplus a_0 (b_3 \oplus b_0) \\
c_2 &= a_2 b_3 \oplus a_3 (b_2 \oplus b_0) \oplus a_4 (b_0 \oplus b_1) \oplus a_0 (b_3 \oplus b_4) \oplus a_1 (b_4 \oplus b_1) \\
c_3 &= a_3 b_4 \oplus a_4 (b_3 \oplus b_1) \oplus a_0 (b_1 \oplus b_2) \oplus a_1 (b_4 \oplus b_0) \oplus a_2 (b_0 \oplus b_2) \\
c_4 &= a_4 b_0 \oplus a_0 (b_4 \oplus b_2) \oplus a_1 (b_2 \oplus b_3) \oplus a_2 (b_0 \oplus b_1) \oplus a_3 (b_1 \oplus b_3)
\end{aligned}$$

or, generally, for $0 \le k \le 4$,

$$c_k = a_k b_{1+k} \oplus a_{1+k}(b_k \oplus b_{3+k}) \oplus a_{2+k}(b_{3+k} \oplus b_{4+k}) \oplus a_{3+k}(b_{1+k} \oplus b_{2+k}) \oplus a_{4+k}(b_{2+k} \oplus b_{4+k}),$$

where indices are of modulo $m = 5$.

Example: $(10011) \cdot (10101) = \alpha^{15} \alpha^{26} = \alpha^{41} = \alpha^{10} = (10001)$ or by another way

$(10011) \cdot (10101) = (c_0, c_1, c_2, c_3, c_4) = (10001)$ because:

$$\begin{aligned}
c_0 &= (1)(0) \oplus (0)(1 \oplus 0) \oplus (0)(0 \oplus 1) \oplus (1)(0 \oplus 1) \oplus (1)(1 \oplus 1) = 1 \\
c_1 &= (0)(1) \oplus (0)(0 \oplus 1) \oplus (1)(1 \oplus 1) \oplus (1)(1 \oplus 0) \oplus (1)(0 \oplus 1) = 0 \\
c_2 &= (0)(0) \oplus (1)(1 \oplus 1) \oplus (1)(1 \oplus 0) \oplus (1)(0 \oplus 1) \oplus (0)(1 \oplus 0) = 0 \\
c_3 &= (1)(1) \oplus (1)(0 \oplus 0) \oplus (1)(0 \oplus 1) \oplus (0)(1 \oplus 1) \oplus (0)(1 \oplus 1) = 0 \\
c_4 &= (1)(1) \oplus (1)(1 \oplus 1) \oplus (0)(1 \oplus 0) \oplus (0)(1 \oplus 0) \oplus (1)(0 \oplus 0) = 1.
\end{aligned}$$

Consider a non-supersingular elliptic curve E: $y^2 + xy = x^3 + \beta x^2 + \beta$ over the finite field $GF(2^5)$ with ONB Type II above.

$E_2 = \{O, (0, \beta^{16}), (\beta^2, \beta), (\beta^2, \beta^{20}), (\beta^3, \beta^{20}), (\beta^3, \beta^{24}), (\beta^4, \beta^{16}), (\beta^4, \beta^{17}), (\beta^5, \beta^4),$
$(\beta^5, \beta^{23}), (\beta^{11}, \beta^{13}), (\beta^{11}, \beta^{18}), (\beta^{12}, \beta^{16}), (\beta^{12}, \beta^{26}), (\beta^{13}, \beta^{11}), (\beta^{13}, \beta^{18}), (\beta^{14}, \beta^7),$

(β^{14}, β^{9}), (β^{17}, β^{13}), (β^{17}, β^{27}), (β^{18}, β^{6}), (β^{18}, β^{19}), (β^{20}, β^{13}), (β^{20}, β^{15}), (β^{22}, β), (β^{22}, β^{18}), (β^{23}, β^{21}), (β^{23}, β^{28}), (β^{24}, β^{3}), (β^{24}, β^{20}), (β^{25}, β^{12}), (β^{25}, β^{24}), (β^{27}, β), (β^{27}, β^{25}), (β^{29}, β^{22}), (β^{29}, β^{24}), (β^{30}, β^{2}), $(\beta^{30}, \beta^{10})\}$.

Its order is 38. The structure of the group is $Z_{n_1} \oplus Z_{n_2}$, where $n_2 \mid \gcd(n_1, 31)$. Since n_1 is a divisor of 38, and 31 is prime; hence $n_2 = 1$ and $E = Z_{38}$, a cyclic group.

References

[AKK96] Tolga Acar, Burton S. Kaliski, Jr. & Çetin K. Koç, Analyzing and comparing Montgomery multiplication algorithms, pp. 26-33, IEEE Micro, 16, 3, June 1996.

[AK98] Tolga Acar & Çetin K. Koç, Montgomery multiplication in $GF(2^k)$, pp. 57-69, Designs, Codes and Cryptography, 14, 1998.

[ARS78] Leonard M. Adleman, Ronald L. Rivest & Adi Shamir, A method for obtaining digital signatures and public-key cryptosystems, pp. 120-126, Communications of the ACM, 21, 2, February 1978.

[A91] V. Afanasyev, On the complexity of finite field arithmetic, pp. 9-12, the 5th Joint Soviet-Swedish International Workshop on Information Theory, Moscow, USSR, January 1991.

[ABMV93] Gordon B. Agnew, Thomas Beth, Ron C. Mullin & Scott A. Vanstone, Arithmetic operation in $GF(2^m)$, pp. 3-13, Journal of Cryptology, 6, 1, 1993.

[AMOV91] Gordon B. Agnew, Ron C. Mullin, I. Onyszchuk & Scott A. Vanstone, An implementation for a fast public-key cryptosystem, pp. 63-79, Journal of Cryptology, 3, 2, 1991.

[AMV90] Gordon B. Agnew, Ron C. Mullin & Scott A. Vanstone, Improved digital scheme based on discrete exponentiation, p. 1024, Electronics Letters, 26, 14, July 1990.

[AMV93] Gordon B. Agnew, Ron C. Mullin & Scott A. Vanstone, An implementation of elliptic curve cryptosystems over F_{2155}, pp. 804-813, IEEE Journal on Selected Areas in Communications, 11, 5, June 1993.

[AO01] Manfred Aigner & Elisabeth Oswald, Randomized addition-subtraction chains as a countermeasure against power attacks, pp. 39-50, Lecture Notes in Computer Science (LNCS) 2162, Workshop on Cryptographic Hardware and Embedded Systems – CHES 2001, Paris, France, Çetin K. Koç, David Naccache & Christof Paar (eds), Springer, 2001.

[AHKKM] Kazumaro Aoki, Fumitaka Hoshino, Kunio Kobayashi, Tetsutaro Kobayashi & Hikaru Morita, Usage of optimal extension fields for elliptic curve cryptosystems, Nippon Telegraph and Telephone (NTT) Laboratories.

[AS98] Kiyomichi Araki & Takakazu Satoh, Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves, pp. 81-92, Commentarii Mathematici Universitatis Sancti Pauli, 47, 1998. Eratta: pp. 211-213, ibid, 48, 1999.

[ABV89] David W. Ash, Ian F. Blake & Scott A. Vanstone, Low complexity normal bases, pp. 191-210, Discrete Applied Mathematics, 25, 1989.

[B01] Harald Baier, Elliptic curve of prime order over optimal extenstion fields for use in cryptography, 2001.

[BB01] Harald Baier & Johannes Buchmann, Efficient construction of cryptographically strong elliptic curves, pp. 191-202, LNCS 1977, Indocrypt 2000, Springer-Verlag, 2000; Technical Report TI-02/01, Technische Universität Darmstadt, 2001.

[BP98] Daniel V. Bailey & Christof Paar, Optimal extension fields for fast arithmetic in public-key algorithms, pp. 472-485, LNCS 1462, Advances in Cryptology – Crypto '98, Santa Barbara, California, Hugo Krawczyk (ed.), Springer-Verlag, 1998.

[BP99] Daniel V. Bailey & Christof Paar, Inversion in optimal extension fields, Conference on the Mathematics of Public Key Cryptography, Andrew Odlyzko, G. Walsh & H. Williams (eds.), Fields Institute for Research in the Mathematical Sciences, Toronto, Canada, June 1999.

[BP00] Daniel V. Bailey & Christof Paar, Efficient arithmetic in finite field extensions with application in elliptic curve cryptography, Journal of Cryptology, 2001.

[BPW00] Daniel V. Bailey, Christof Paar & Adam D. Woodbury, Elliptic curve cryptography on smart cards without coprocessors, the 4th Smart Card Research and Advanced Applications (IFIP CARDIS 2000) Conference, Bristol, United Kingdom, September 2000.

[BK98] R. Balasubramanian & Neal Koblitz, The improbability that an elliptic curve has sub-exponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm, pp. 141-145, Journal of Cryptology, 11, 2, Spring 1998.

[Be01] Antonio Bellezza, Countermeasures against side-channel attacks for elliptic curve cryptosystems, November 2001.

[BFT96] Mohammed Benaissa, Sebastian T. J. Fenn & David Taylor, $GF(2^m)$ multiplication and division over the dual basis, pp. 319-327, IEEE Transactions on Computers, 45, 3, March 1996.

[BFT96a] Mohammed Benaissa, Sebastian T. J. Fenn & David Taylor, Finite field inversion over the dual base, pp. 134-136, IEEE Transactions on VLSI Systems, 4, March 1996.

[BS91] Thomas Beth & F. Schaefer, Non-supersingular elliptic curves for public key cryptosystems, pp. 252-266, LNCS 547, Advances in Cryptology – Eurocrypt '91, Brighton, United Kingdom, Donald W. Davies (ed.), Springer-Verlag, 1991.

[BMM00] Ingrid Biehl, Bernd Meyer & Volker Müller, Differential fault attacks on elliptic curve cryptosystems, pp. 131-146, LNCS 1880, Advances in Cryptology – Crypto 2000, Santa Barbara, California, Mihir Bellare (ed.), Springer-Verlag, 2000.

[BB79] G. R. Blakley & I. Borosh, Rivest-Shamir-Adleman public key cryptosystems do not always conceal messages, pp. 169-178, Computers and Mathematics with Applications, 5, 1979.

[B96] Daniel Bleichenbacher, Generating ElGamal signatures without knowing the secret key, LNCS 1070, Advances in Cryptology – Eurocrypt '96, Saragossa, Spain, Ueli M. Maurer (ed.), Springer-Verlag, 1996.

[B97] Daniel Bleichenbacher, On the security of the KMOV public key cryptosystems, pp. 235-248, LNCS 1294, Advances in Cryptology – Crypto '97, Santa Barbara, California, Burt S. Kaliski, Jr. (ed.), Springer, 1997.

[BJQ97] Daniel Bleichenbacher, Marc Joye, and Jean-Jacques Quisquater, A new and optimal chosen-message attack on RSA-type cryptosystems, pp. 302-313, International Conference on Information and Communications Security – ICICS '97, LNCS 1334, Yongfei Han, Tatsuaki Okamoto & S. Qing (eds.), Springer-Verlag, 1997.

[B98] Dan Boneh, The decision Diffie-Hellman problem, pp. 48-63, LNCS 1423, Algorithmic Number Theory, the 3rd International Symposium, ANTS-III, Portland, Oregon, Joe P. Buhler (ed.), June 1998.

[BL96] Dan Boneh & Richard J. Lipton, Searching for elements in black box fields and applications, pp. 283-297, LNCS 1109, Advances in Cryptology – Crypto '96, Santa Barbara, California, Neal Koblitz (ed.), Springer-Verlag, 1996.

[BS01] Dan Boneh & Igor E. Shparlinski, On the unpredictability of bits of the elliptic curve Diffie-Hellman scheme, p. 201 ff, LNCS 2139, Advances in Cryptology – Crypto 2001, Santa Barbara, California, Joe Kilian (ed.), Springer-Verlag, 2001.

[BV96] Dan Boneh & Ramarathnam Venkatesan, Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes, pp. 129-142, LNCS 1109, Advances in Cryptology – Crypto '96, Santa Barbara, California, Neal Koblitz (ed.), Springer-Verlag, 1996.

[BC90] Jurjen Bos & Matthijs Coster, Addition chain heuristics, pp. 400-407, LNCS 435, Advances in Cryptology – Crypto '89, Santa Barbara, California, Gilles Brassard (ed.), Springer-Verlag, 1990.

[BGVVW96] Antoon Bosselaers, P. de Gersem, Servaas Vandenberghe, Joos Vandewalle & Erik de Win, A fast software implementation for arithmetic operations in $GF(2^n)$, pp. 65-76, LNCS 1163, Advances in Cryptology – Asiacrypt '96, Kyongju, Korea, Kwangjo Kim & Tsutomu Matsumoto (eds.), Springer-Verlag, 1996.

[BS95] Colin Boyd & Andrew Smith, An elliptic curve analogue of McCurley's key agreement scheme, the 5th IMA conference on Cryptography and Coding, Springer-Verlag, pp. 150-157, 1995.

[B87] David M. Bressoud, *Factorization and primality testing,* Undergraduate Texts in Mathematics, Springer, 1987.

[BGMW93] Ernest F. Brickell, Daniel M. Gordon, Kevin S. McCurley & David B. Wilson, Fast exponentiation with precomputation, pp. 200-207, LNCS 658, Advances in Cryptology – Eurocrypt '92, Balatonfüred, Hungary, Rainer A. Rueppel (ed.), Springer-Verlag, 1993.

[BD85] Ernest F. Brickell & J. DeLaurentis, An attack on a signature scheme proposed by Okamoto and Shiraishi, pp. 28-32, LNCS 218, Advances in Cryptology – Crypto '85, Santa Barbara, California, Hugh C. Williams (ed.), Springer-Verlag, 1986.

[B00] Daniel R. L. Brown, The exact security of ECDSA, preprint, 2000.

[BHHM01] Micheal Brown, Darrel Hankerson, Julio López Hernandez & Alfred Menezes, Software implementation of the NIST elliptic curves over prime fields, pp. 250-265, LNCS 2020, Topics in Cryptology – Cryptographer's Track – RSA Conference 2001, David Naccache (ed.), Springer-Verlag, 2001.

[BM91] Johannes Buchmann & Volker Müller, Computing the number of points of elliptic curves over finite fields, the 1991 International Symposium on Symbolic & Algebraic Computations, Bonn, 1991.

[BMS95] Johannes Buchmann, Volker Müller & Victor Shoup, Distributed computation of the number of points on an elliptic curve over a finite prime field, Technical Report 03/95 SFB 124-TP D5, University of Saarland, 1995.

[C52] L. Carlitz, Primitive roots in finite field, pp. 373-382, Transactions of the American Mathematical Society, 73, 1952.

[C52a] L. Carlitz, Some problems involving primitive roots in a finite field, pp. 314-318 and p. 618, Proceedings of the National Academy Science of the U.S.A., 38, 1952.

[CGH01] D. Catalano, Rosario Gennaro & N. Howgrave-Graham, The bit security of Paillier's encryption scheme and its applications, pp. 229-243, LNCS 2045, Advances in Cryptology – Eurocrypt 2001, Innsbuck, Austria, Birgit Pfitzmann (ed.), Springer-Verlag, 2001.

[CTT94] Jinhui Chao, K. Tanada & Shigeo Tsujii, Design of elliptic curves with controllable lower boundary of extension degree for reduction attacks, pp. 50-55, LNCS 839, Advances in Cryptology – Crypto '94, Santa Barbara, California, Yvo G. Desmedt (ed.), Springer-Verlag, 1994.

[CL97] S. K. Chua & S. Ling, A Rabin-type scheme on $y^2 \equiv x^3 + bx^2 \pmod{n}$, LNCS 1276, the 3rd Annual International Computing and Combinatorics Conference (COCOON '97), T. Jiang & D. T. Lee (eds.), Springer-Verlag, 1997.

[CMO97] Henri Cohen, Atsuko Miyaji & Takatoshi Ono, Efficient elliptic curve exponentiation, pp. 282-290, LNCS 1334, International Conference on Information and Communications Security – ICICS '97, Yongfei Han, Tatsuaki Okamoto & S. Qing (eds.), Springer-Verlag, 1997.

[CMO98] Henri Cohen, Atsuko Mijyji & Takatoshi Ono, Efficient elliptic curve exponentiation using mixed coordinates, pp. 51-65, LNCS 1514, Advances in Cryptology – Asiacrypt '98, Beijing, China, Kazuo Ohta & Dingyi Pei (eds.), Springer-Verlag, 1999.

[COS86] Don Coppersmith, Andrew M. Odlyzko & Richard Schroeppel, Discrete logarithms in $GF(p)$, pp. 1-15, Algorithmica, 1, 1986.

[C99] Jean-Sébastien Coron, Resistance against differential power analysis attacks for elliptic curve cryptosystems, pp. 292-302, LNCS 1717, Workshop on Cryptographic Hardware and Embedded Systems – CHES '99, Çetin K. Koç & Christof Paar (eds), Springer, 1999.

[C94] Jean-Marc Couveignes, Quelques calculs en théorie des nombres, Thèse, Université de Bordeaux I, July 1994.

[C96] Jean-Marc Couveignes, Computing l-isogenies with the p-torsion, pp. 59-66, LNCS 1122, Algorithmic Number Theory, the 2nd International Symposium, ANTS-II, Talence, France, May 1996, Henri Cohen (ed.), Springer-Verlag, 1996.

[CDM96] Jean-Marc Couveignes, L. Dewaghe & François Morain, Isogeny cycles and the Schoof-Elkies-Atkin algorithm, Research report LIX/RR/96/03, École Polytechnique, Laboratoire D'informatique, 1996.

[CM94] Jean-Marc Couveignes & François Morain, Schoof's algorithm and isogeny cycles, pp. 43-58, LNCS 877, Algorithmic Number Theory, the 1st International Symposium, ANTS-I, Ithaca, Springer-Verlag, 1994.

[C92] Richard E. Crandall, U.S. Patent No. 5,159,632, Method and apparatus for public key exchange in a cryptographic system, 27 October 1992.

[DJ01] Ivan B. Damgård & M. J. Jurik, A generalisation, a simplification and some applications of Paillier's probabilistic public-key system, pp. 365-382, LNCS 1992, Public Key Crypography 2001, Korea, Kwangjo Kim (ed.), Springer-Verlag, 2001.

[D69] H. Davenport, Bases for finite fields, Journal of London Mathematical Society, 43, 1968, pp.21-39 and 44, p. 378, 1969.

[D94] N. Demytko, A new elliptic curve based analogue of RSA, pp. 40-49, LNCS 765, Advances in Cryptology – Eurocrypt '93, Lofthus, Norway, Tor Helleseth (ed.), Springer-Verlag, 1994.

[DHRT88] L. J. Deutsch, I. S. Hsu, Irving S. Reed & T. K. Truong, A comparison of VLSI architecture of finite field multipliers using dual, normal or standard bases, pp. 735-739, IEEE Transactions on Computers, 37, 6, June 1988.

[DORSTW85] L. J. Deutsch, James K. Omura, Irving S. Reed, H. M. Shao, T. K. Truong & Charles C. Wang, VLSI architectures for computing multiplications and inverses in $GF(2^m)$, pp. 709-716, IEEE Transactions on Computing, C-34, 8, 1985.

[D98] L. Dewaghe, Remarks on the Schoof-Elkies-Atkin algorithm, pp. 1247-1252, Mathematics of Computation, 67, 223, July 1998.

[DS92] Andreas Dietel & Jorg Sauerbrey, Resource requirements for the application of addition chains in modulo exponentiation, pp. 174-182, LNCS 658, Advances in Cryptology – Eurocrypt '92, Balatonfüred, Hungary, Rainer A. Rueppel (ed.), Springer-Verlag, 1993.

[DH76] Whitfield Diffie & Martin E. Hellman, New directions in cryptography, pp. 644-654, IEEE Transactions in Information Theory, 22, 1976.

[DLS81] Peter Downey, Benton Leong & Ravi Sethi, Computing sequences with addition chains, pp. 638-696, SIAM Journal of Computing 3, 1981.

[EK94] Ömer Eğecioğlu & Çetin K. Koç, Exponentiation using canonical recoding, pp. 407-417, Theoretical Computer Science 129, 1994.

[E85] Taher ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, pp. 469-472, IEEE Transactions in Information Theory, IT-31, 1985.

[E98] Noam D. Elkies, Elliptic and modular curves over finite fields and related computational issues, pp. 21-76, Computational Perspectives on Number Theory, D. A. Buell & J. T. Teitelbaum (eds.), AMS/International Press, 1998.

[FaP97] John L. Fan & Christof Paar, On efficient inversion in tower fields of charatersitics two, 1997 IEEE International Symposium on Information Theory (ISIT), Ulm, Germany, 1997.

[FP97] R. Flassenberg & Sachar Paulus, Sieving in function fields, the ECDLP workshop '97, University of Waterloo, Ontario, Canada, 1997.

[FPR98] Peter Fleischmann, Christof Paar & P. Roelse, Efficient multiplier architectures for Galois fields $GF(2^{4n})$, pp. 162-170, IEEE Transactions on Computers, 47, 2, February 1998.

[FPS99] Peter Fleischmann, Christof Paar & Pedro Soria-Rodriguez, Fast arithmetic for public-key algorithms in Galois fields with composite exponents, pp. 1025-1034, IEEE Transactions on Computers, 48, 10, October 1999.

[FGH00] Mireille Fouquet, Pierrick Gaudry & Robert Harley, An extension of Satoh's algorithm and its implementation, 2000.

[FGH01] Mireille Fouquet, Pierrick Gaudry & Robert Harley, Finding secure curves with the Satoh-FGH algorithm and an early-abort strategy, pp. 14 ff, LNCS 2045, Advances in Cryptology – Eurocrypt 2001, Innsbuck, Austria, Birgit Pfitzmann (ed.), Springer-Verlag, 2001.

[FFO93] Atsushi Fujioka, Eiichiro Fujisaki & Tatsuaki Okamoto, An efficient digital scheme based on an elliptic curve over the ring Z_N, pp. 54-65, LNCS 740, Advances in Cryptology – Crypto '92, Santa Barbara, California, Ernest F. Brickell (ed.), Springer-Verlag, 1993.

[G99] Steven D. Galbraith, Constructing isogenies between elliptic curves over finite fields, pp. 118-138, LMS Journal of Computation and Mathematics, 2, 1999.

[GLV00] Robert Gallant, Robert Lambert & Scott A. Vanstone, Improving the parallelized Pollard lambda search on binary anomalous curves, pp. 1699-1705, Mathematics of Computation, 69, 232, 2000.

[GLV01] Robert Gallant, Robert Lambert & Scott A. Vanstone, Faster point multiplication on elliptic curves with efficient endomorphisms,), p. 190 ff, LNCS 2139, Advances in Cryptology – Crypto 2001, Santa Barbara, California, Joe Kilian (ed.), Springer-Verlag, 2001.

[GL92] Shuhong Gao & Hendrik W. Lenstra, Jr., Optimal normal bases, pp. 315-323, Designs, Codes and Cryptography, 2, 1992.

[GG90] W. Geiselmann & Dieter Gollmann, VLSI design for exponentiation in $GF(2^n)$, pp. 398-405, LNCS 453, Advances in Cryptology – Auscrypt '90, Sydney, Australia, Josef Pieprzyk & Jennifer Seberry (eds.), Springer-Verlag, 1990.

[GTV88] M. Girault, P. Toffin & B. Vallée, How to break Okamoto's cryptosystem by reducing lattice bases, pp. 281-291, LNCS 330, Advances in Cryptology – Eurocrypt '88, Davos, Switzerland, C. Günther (ed.), Springer-Verlag, 1988.

[GHM96] Dieter Gollmann, Y. Han & Chris J. Mitchell, Redundant integer representations and fast exponentiation, pp. 135-151, Designs, Codes and Cryptography, 7, 1996.

[G93] Daniel M. Gordon, Discrete logarithm in $GF(p)$ using the number field sieve, pp. 124-138, SIAM Journal on Discrete Mathematics, 6, 1993.

[G98] Daniel M. Gordon, A survey of fast exponentiation methods, pp. 129-146, Journal of Algorithms, 27, 1998.

[G02] Steven D. Galbraith, Elliptic curve Paillier schemes, pp.129-138, Journal of Cryptography, 15, 2, Spring 2002.

[GT74] D. H. Green & I. S. Taylor, Irreducible polynomials over composite Galois fields and their applications in coding techniques, pp. 935-939, Proceedings of IEE, 121, September 1974.

[G97] Jorge Guajardo, Efficient algorithms for elliptic curve cryptosystems, Master Thesis, ECE Department, Worcester Polytechnic Institute, Worcester, Christof Paar (advisor), May 1997.

[GP97] Jorge Guajardo & Christof Paar, Efficient algorithms for elliptic curve cryptosystems, pp. 342-356, LNCS 1294, Advances in Cryptology – Crypto '97, Santa Barbara, California, Burt S. Kaliski, Jr. (ed.), Springer, 1997.

[GP98] Jorge Guajardo & Christof Paar, Fast inversion in composite Galois fields $GF((2^n)^m)$, 1998 IEEE International Symposium on Information Theory (ISIT), MIT, Cambridge, Massachussetts, 1998.

[GP01] Jorge Guajardo & Christof Paar, Itoh-Tsujii inversion in standard basis and its application in cryptography and codes, to appear in Designs, Codes and Cryptography, 2001.

[HHM00] Darrel Hankerson, Julio L. Hernandez & Alfred Menezes, Software implementation of elliptic curve cryptography over binary fields, LNCS 1965, Workshop on Cryptographic Hardware and Embedded Systems – CHES 2000, Çetin K. Koç & Christof Paar (eds.), Springer-Verlag, 2000.

[HMV93] Greg Harper, Alfred J. Menezes & Scott A. Vanstone, Public-key cryptosystems with very small key lengths, pp. 163-173, LNCS 658, Advances in Cryptology – Eurocrypt '92, Balatonfüred, Hungary, Rainer A. Rueppel (ed.), Springer-Verlag, 1993.

[HW98] M. Anwarul Hasan & Huapeng Wu, Low complexity bit-parallel multipliers for a class of finite fields, pp. 883-887, IEEE Transactions on Computers, 47, 8, August 1998.

[H85] Johan Håstad, On using RSA with low exponent in a public key network, pp. 403-408, LNCS 218, Advances in Cryptology – Crypto '85, Santa Barbara, California, Hugh C. Williams (ed.), Springer-Verlag, 1986.

[HK94] J. He & T. Kiesier, Enhancing the security of ElGamal's signature scheme, pp. 249-252, IEE Proceedings of Computing and Digital Technique, 141, 4, July 1994.

[HP78] Martin E. Hellman & Stephen C. Pohlig, An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance, pp. 106-110, IEEE Transactions on Information Theory, IT-24, January 1978.

[HMP94] P. Horster, M. Michels & H. Petersen, Meta-ElGamal signature schemes, pp. 96-107, the 2nd ACM Conference on Computer and Communications Security, ACM Press, New York, November 1994.

[HMP94a] P. Horster, M. Michels & H. Petersen, Generalized ElGamal signatures for one message block, Technical Report TR-94-3, University of Technology Chemnitz-Zwickau, May 1994.

[HKKM99] Fumitaka Hoshino, Kunio Kobayashi, Tetsutaro Kobayashi & Hikaru Morita, Fast elliptic curve algorithm combining Frobenius map and table reference to adapt to higher characteristic, pp. 176-189, LNCS 1592, Advances in Cryptology – Eurocrypt '99, Prague, Czech Republic, Jacques Stern (ed.), Springer-Verlag, 1999.

[IMT86] Hideki Imai, Tsutomu Matsumoto & Y. Takashima, On seeking smart public-key distribution systems, pp. 99-106, Transactions of the IECE of Japan, E69, 1986.

[IZ98] Hideki Imai & Yuliang Zheng, Efficient signcryption schemes on elliptic curves, IFIP/SEC '98, the 14th Interantional Information Security conference, Vienna & Budapest, August 1998.

[IM85] Kyoki Imamura & M. Morii, Two classes of finite fields which have no self-complementary normal bases, IEEE International Symposium of Information Theory, England, 1985.

[IR90] Kenneth Ireland & Michael Rosen, *A classical introduction to modern number theory*, Graduate Texts in Mathematics, 84, Springer-Verlag, 1990.

[I91] Toshiya Itoh, Characterization for a family of infinitely many irreducible equally spaced polynomials, pp. 273-277, Information Processing Letters, 37, 5, 1991.

[ITT86] Toshiya Itoh, O. Teechai & Shigeo Tsujii, A fast algorithm for computing multiplicative inverses in $GF(2^t)$, pp. 31-36, Journal of the Society for Electronic Communication, Japan, 44, 1986.

[IT88] Toshiya Itoh & Shigeo Tsujii, A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases, pp. 171-177, Information and Computation Journal, 78, 1988.

[JM89] J. Jedwab & Chris J. Mitchell, Minimum weight modified signed-digit representations and fast exponentiation, pp. 1171-1172, Electronics Letters, 25, 17, August 1989.

[JL] Antoine Joux & Reynald Lercier, "Chinese & match", an alternative to Atkin's "Match and sort" method used in the SEA algorithm, Mathematics of Computation, to appear.

[JQ95] Marc Joye & Jean-Jacques Quisquater, Note on the preliminary version of the Meyer-Müller's cryptosystem, Crypto Group Technical Report CG-1996/2, Université Catholique de Louvain, 1995.

[JQ95a] Marc Joye & Jean-Jacques Quisquater, On the cryptosystems of Chua and Ling, Crypto Group Technical Report CG-1997/4, Université Catholique de Louvain, 1995.

[JQ96] Marc Joye & Jean-Jacques Quisquater, Reducing elliptic curve cryptosystem of Meyer-Müller, Crypto Group Technical Report CG-1996/4, Université Catholique de Louvain, 1996. Other title: Reducing elliptic curve cryptosystem of Meyer-Müller to the cryptosystem of Rabin-Williams.

[JQT97] Marc Joye, Jean-Jacques Quisquater & Tsuyoshi Takagi, How to choose secret parameters for RSA-type cryptosystems over elliptic curves, Technical Report TI-35/97, Technische Universität Darmstadt, November 1997.

[JT01] Marc Joye & Christophe Tymen, Protections against differential analysis for elliptic curve cryptography – an algebraic approach, pp. 377-390, LNCS 2162, Workshop on Cryptographic Hardware and Embedded Systems – CHES 2001, Paris, France, Çetin K. Koç, David Naccache & Christof Paar (eds), Springer, 2001.

[JMV90] Dieter Jungnickel, Alfred J. Menezes & Scott A. Vanstone, On the number of self-dual bases of $GF(q^m)$ over $GF(q)$, pp. 23-29, Proceedings of the American Mathematical Society, 109, 1, May 1990.

[K91] Burt S. Kaliski, Jr., One-way permutations on elliptic curves, pp. 187-199, Journal of Cryptology, 3, 3, 1991.

[K97] Burt S. Kaliski, Jr., A chosen message attack on Demytko's elliptic curve cryptosystems, pp. 71-72, Journal of Cryptology, 10, 1, 1997.

[KL98] Burt S. Kaliski, Jr. & Moses Liskov, pending U.S. Patent application serial No. 09/195,346, Efficient finite field basis conversion involving a dual basis, November 1998.

[KY98] Burt S. Kaliski, Jr. & Yiqun Lisa Yin, Storage-efficient finite field basis conversion, the 5th Annual Workshop on Selected Areas in Cryptography – SAC '98.

[KY98p] Burt S. Kaliski, Jr. & Yiqun Lisa Yin, U.S. Patent No. 5,854,759, Methods and apparatus for efficient finite field basis conversion, December 1998.

[KO63] A. Karatsuba & Y. Ofman, Multiplication of multidigit numbers on automata, Soviet Physics (Doklady Akademii Nauk, USSR), 1962; English translation, 7, 7, pp. 595-596, 1963.

[KM89] M. Kasahara & M. Morii, Efficient construction of gate circuit for computing multiplicative inverses over $GF(2^m)$, pp. 37-42, IEICE Transactiions, E-72, 1, 1989.

[KLL98] D. S. Kim, E. J. Lee, P. J. Lee, Speed-up of F_{p^m} arithmetic for elliptic curve cryptosystems, International Conference on Information and Communications Security – ICICS '98, Springer-Verlag, 1998.

[Kn99] E. Knudsen, Elliptic scalar multiplication using point halving, pp. 135-149, LNCS 1716, Advances in Cryptology – Asiacrypt '99, Singapore, Kwok-Yan Lam, Eiji Okamoto & Chaoping Xing (eds.), Springer-Verlag, 1999.

[K87] Neal Koblitz, Elliptic curve cryptosystems, pp. 203-209, Mathematics of Computation, 48, 177, January 1987.

[K90] Neal Koblitz, Constructing elliptic curve cryptosystems in characteristic 2, pp. 156-167, LNCS 537, Advances in Cryptology – Crypto '90, Santa Barbara, California, Alfred J. Menezes & Scott A. Vanstone (eds.), Springer-Verlag, 1991.

[K92] Neal Koblitz, CM-curves with good cryptographic properties, pp. 279-287, LNCS 576, Advances in Cryptology – Crypto '91, Santa Barbara, California, Joan Feigenbaum (ed.), Springer-Verlag, 1992.

[K98] Neal Koblitz, An elliptic curve implementation of the finite field digital signature algorithm, pp. 327-337, LNCS 1462, Advances in Cryptology – Crypto '98, Santa Barbara, California, Hugo Krawczyk (ed.), Springer-Verlag, 1998.

[KMV96] Neal Koblitz, Alfred J. Menezes & Scott A. Vanstone, The state of elliptic curve cryptography, October 1996, to appear in Designs, Codes and Cryptography.

[Ko96] Paul C. Kocher, Timing attacks on implementations of Diffie-Heelman, RSA, DSS and other systems, LNCS 1109, Advances in Cryptology – Crypto '96, Santa Barbara, California, Neal Koblitz (ed.), Springer-Verlag, 1996.

[Kc91] Çetin K. Koç, High-radix and bit recoding techniques for modular exponentiation, pp. 139-156, International Journal of Computer Mathematics, 40, 1991.

[Kc95] Çetin K. Koç, Analysis of sliding window techniques for exponentiation, pp. 17-24, Computers and Mathematics with Applications, 30, 10, November 1995.

[Ky95] Kenji Koyama, Fast RSA-type schemes based on singular cubic curves $y^2 + axy = x^3 \pmod{n}$, pp. 329-340, LNCS 921, Advances in Cryptology – Eurocrypt '95,

Saint-Malo, France, Louis C. Guillou & Jean-Jacques Quisquater (eds.), Springer-Verlag, 1995.

[KK94] Kenji Koyama & Hidenori Kuwakado, Efficient cryptosystems over elliptic curves based on a product of form-free primes, pp. 1309-1318, IEICE Transactions of Fundamentals on Electronic Communications Computer Science, E77-A, 8, 1994.

[KK94a] Kenji Koyama & Hidenori Kuwakado, Security of RSA-type cryptosystems over elliptic curves against the Håstad attack, pp. 1834-1844, Electronics Letters, 30, 22, 1994. Other version: Kenji Koyama & Hidenori Kuwakado, On the security of RSA-type schemes over cubic curves against the Håstad attack, pp. 23-30, Technical Report of IEICE, ISEC94-10, 1994.

[KMOV92] Kenji Koyama, Ueli M. Maurer, Tatsuaki Okamoto & Scott A. Vanstone, New public-key scheme based on elliptic curves over the ring Z_n, pp. 252-266, LNCS 576, Advances in Cryptology – Crypto '91, Santa Barbara, California, Joan Feigenbaum (ed.), Springer-Verlag, 1992.

[KO97] Kenji Koyama & Tatsuaki Okamoto, NTT's research & development of public-key cryptosystems, Public Key Solution conference '97.

[KT92] Kenji Koyama & Yukio Tsuruoka, Speeding up elliptic cryptosystems by using a signed binary window method, pp. 345-357, LNCS 740, Advances in Cryptology – Crypto '92, Santa Barbara, California, Ernest F. Brickell (ed.), Springer-Verlag, 1992.

[KOT95] Kaoru Kurosawa, Koji Okada & Shigeo Tsujii, Low exponent attack against elliptic curve RSA, pp. 376-383, LNCS 917, Advances in Cryptology – Asiacrypt '94, Wollongong, Australia, Josef Pieprzyk & Reihaneh Safavi-Naini (eds.), Springer-Verlag, 1995. Other version: Kaoru Kurosawa & Shigeo Tsujii, Low exponent attack against elliptic curve RSA, pp. 11-17, Technical Report of IEICE, ISEC94-2, 1994.

[LMQSV98] Laurie Law, Alfred J. Menezes, Minghua Qu, Jerome Solinas & Scott A. Vanstone, An efficient protocol for authenticated key agreement, Technical Report CORR 98-05, University of Waterloo, Ontario, Canada, March 1998.

[LZ94] Georg-Johann Lay & Horst-Günter Zimmer, Constructing elliptic curves with given group order over large finite fields, pp. 250-263, LNCS 877, Algorithmic Number Theory, the 1st International Symposium, ANTS-I, Ithaca, Springer-Verlag, 1994.

[LL98] Chang-Hyi Lee & Jong-In Lim, A new aspect of dual basis for efficient field arithmetic, 1998.

[LL94] Pil Joong Lee & Chae Hoon Lim, More flexible exponentiation with precomputation, pp. 95-107, LNCS 839, Advances in Cryptology – Crypto '94, Santa Barbara, California, Yvo G. Desmedt (ed.), Springer-Verlag, 1994.

[L77] J. van Leeuwen, An extension of Hansen's theorem for star chains, pp. 203-207, Journal Reine Angew. Math. 295, 1977.

[LMMS94] Frank Lehmann, Ueli M. Maurer, Volker Müller & Victor Shoup, Counting the number of points on elliptic curves over finite fields of characteristic greater than three, pp. 60-70, LNCS 877, Algorithmic Number Theory, the 1st International Symposium, ANTS-I, Ithaca, Springer-Verlag, 1994.

[LS80] Abraham Lempel & Gadiel Seroussi, Factorization of symmetric matrices and trace-orthogonal bases in finite fields, pp.758-767, SIAM Journal of Computing, 9, 4, November 1980.

[LW88] Abraham Lempel & M. J. Weinberger, Self-complementary normal bases in finite fields, pp. 193-198, SIAM Journal of Discrete Mathematics 1, 1988.

[LL90] Arjen K. Lenstra & Hendrik W. Lenstra, Jr., Algorithm in number theory, pp. 673-715, *Handbook of Theoretical Computer Science*, J. van Leeuwen (ed.), MIT Press, Cambridge, Massachusetts, 1990.

[LLMP90] Arjen K. Lenstra, Hendrik W. Lenstra, Jr., Mark Manasse & John M. Pollard, The number field sieve, pp. 564-572, the 22nd Annual ACM Symposium on Theory of Computing, 1990.

[LV00] Arjen K. Lenstra & Eric R. Verheul, Selecting cryptographic key sizes, Journal of Cryptography, 2000; the 3rd International workshop on Practice and Theory in Public Key Crypography – PKC 2000, LNCS 1751, Hideki Imai & Yuliang Zheng (eds.), Springer-Verlag, 2000; CCE Quarterly Journal 3, pp. 3-10, 1999.

[LS87] Hendrik W. Lenstra, Jr. & René Schoof, Primitive normal bases for finite fields, pp. 217-231, Mathematics of Computation, 48, 177, January 1987.

[L96] Reynald Lercier, Computing isogenies in F_{2^m}, pp. 197-212, LNCS 1122, Algorithmic Number Theory, the 2nd International Symposium, ANTS-II, Talence, France, Henri Cohen (ed.), Springer-Verlag, 1996.

[L97] Reynald Lercier, Finding good random elliptic curves for cryptosystems defined over F_{2^m}, pp. 379-392, LNCS 1233, Advances in Cryptology – Eurocrypt '97, Konstanz, Germany, Walter Fumy (ed.), Springer-Verlag, 1997.

[L97a] Reynald Lercier, Algorithmique des courbes elliptiques dans les corps finis, Thèse, École Polytechnique, Laboratoire D'informatique, June 1997.

[LM95] Reynald Lercier & François Morain, Counting points on elliptic curves over F_{p^n} using Couveignes' algorithm, Research report LIX/RR/95/00, École Polytechnique, Laboratoire D'informatique, 1995.

[LM95a] Reynald Lercier & François Morain, Counting the number of points on elliptic curves over finite fields: Strategies and performances, pp. 79-94, LNCS 921, Advances in Cryptology – Eurocrypt '95, Saint-Malo, France, Louis C. Guillou & Jean-Jacques Quisquater (eds.), Springer-Verlag, 1995.

[LM97] Reynald Lercier & François Morain, Algorithms for computing isogenies between elliptic curves, Computational Perspectives on Number Theory, 1997.

[MS77] F. J. MacWilliams & Neil J. A. Sloane, *The theory of error-correcting codes*, North-Holland, Amsterdam, 1977.

[MNS01] E. El Mahassni, Phong Q. Nguyen & Igor E. Shparlinski, The insecuriry of Nyberg-Rueppel and other DSA-like signature schemes with partially known nonces, Workshop on Lattices and Cryptography, Boston, MA, 2001.

[MOS99] M. Mambo, Eiji Okamato & H. Sakazaki, ID-based key distribution system over an elliptic curve, pp. 215-233, Contemporary Mathematics, 225, 1999; the 4th International Conference on Finite Fields, 1999.

[MO86] James L. Massey & James K. Omura, U.S. Patent No. 4,567,600, Method and apparatus for maintaining the privacy of digital messages conveyed by public transmission, 28 January 1986.

[Mc86] D. P. McCarthy, Effect of improved multiplication efficiency on exponentiation algorithms derived from addition chains, Mathematics of Computation, 46, 174, pp. 603-608, April 1986.

[Mc88] Kevin S. McCurley, A key distribution system equivalent to factoring, pp. 95-105, Journal of Cryptology, 1, 2, 1988.

[M86] J. F. Mestre's Formules explicites at minoration de conducteurs de variete algebriques, Compositio Math., 58, pp. 209-232, 1986.

[MS93] Willi Meier & Othmar Staffelbach, Efficient multiplication on certain non-supersingular elliptic curves, pp. 333-344, LNCS 740, Advances in Cryptology – Crypto '92, Santa Barbara, California, Ernest F. Brickell (ed.), Springer-Verlag, 1993.

[MOV93] Alfred J. Menezes, Tatsuaki Okamoto & Scott A. Vanstone, Reducing elliptic curve logarithms to logarithms in a finite field, pp. 1639-1646, IEEE Transactions on Information Theory, 39, 1993; the 23rd ACM Symposium on the Theory of Computing, 1991.

[MQV95] Alfred J. Menezes, Minghua Qu & Scott A. Vanstone, Some new key agreement protocols providing mutual implicit authentication, pp. 22-32, Selected Areas in Cryptology – SAC '95.

[MV90] Alfred J. Menezes & Scott A. Vanstone, The implementation of elliptic curve cryptosystems, pp. 2-13, LNCS 453, Advances in Cryptology – Auscrypt '90, Sydney, Australia, Josef Pieprzyk & Jennifer Seberry (eds.), Springer-Verlag, 1990.

[MV90a] Alfred J. Menezes & Scott A. Vanstone, Isomorphism classes of elliptic curves over finite fields of characteristic 2, pp. 135-153, Utilitas Mathematica, 38, 1990.

[MV93] Alfred J. Menezes & Scott A. Vanstone, Elliptic curve cryptosystems and their implementation, pp. 209-224, Journal of Cryptology, 6, 4, 1993.

[MVZ93] Alfred J. Menezes, Scott A. Vanstone & Robert J. Zuccherato, Counting points on elliptic curves over F_{2^m}, pp. 407-420, Mathematics of Computation 60, 201, January 1993.

[MM96] Bernd Meyer & Volker Müller, A public key cryptosystem based on elliptic curves over Z/nZ equivalent to factoring, pp. 49-59, LNCS 1070, Advances in Cryptology – Eurocrypt '96, Saragossa, Spain, Ueli M. Maurer (ed.), Springer-Verlag, 1996.

[Mi85] Victor S. Miller, Use of elliptic curves in cryptography, pp. 417-426, LNCS 218, Advances in Cryptology – Crypto '85, Santa Barbara, California, Hugh C. Williams (ed.), Springer-Verlag, 1986.

[Mi98] Victor S. Miller, Elliptic curves and their uses in cryptography, the 2nd workshop on Elliptic Curve Cryptography (ECC '98), University of Waterloo, Ontario, Canada, 1998.

[MPWW98] Serge Mister, Bart Preneel, Michael Wiener & Erik de Win, On the performance of signature schemes based on elliptic curves, pp. 252-266, LNCS

1423, Algorithmic Number Theory, the 3rd International Symposium, ANTS-III, Portland, Oregon, Joe P. Buhler (ed.), June 1998.

[MS89] Chris J. Mitchell & A. Selby, Algorithms for software implementation of RSA, pp. 166-170, IEE Proceedings, 136, 3, 1989.

[Mi91] Atsuko Miyaji, On ordinary elliptic curve cryptosystems, pp. 50-55, LNCS 739, Advances in Cryptology – Asiacrypt '91, Fujiyoshida, Japan, Hideki Imai, Ronald L. Rivest & Tsutomu Matsumoto (eds.), Springer-Verlag, 1993.

[Mi93] Atsuko Miyaji, Elliptic curves over F_p suitable for cryptosystems, pp. 479-491, LNCS 718, Advances in Cryptology – Auscrypt '92, Jennifer Seberry & Yuliang Zheng (eds.), Springer-Verlag, 1993.

[M85] Peter L. Montgomery, Modular multiplication withour trial division, pp. 519-521, Mathematics of Computation, 44, 170, April 1985.

[M87] Peter L. Montgomery, Speeding the Pollard and elliptic curve methods of factorization, pp. 243-264, Mathematics of Computation, 48, 177, January 1987.

[Mo95] François Morain, Calcul du nombre de points sur une courbe elliptique dans un corps fini: aspects algorithmiques, pp. 255-282, Journal de Théorie des Nombres de Bordeaux, 7, 1995.

[Mo91] François Morain, Building cyclic elliptic curve modulo large primes, pp. 328-336, LNCS 547, Advances in Cryptology – Eurocrypt '91, Brighton, United Kingdom, Donald W. Davies (ed.), Springer-Verlag, 1991.

[MO90] François Morain & Jorge Olivos, Speeding up the computations on an elliptic curve using addition-subtraction chains, pp. 531-544, Informatique théorique et Applications (Theoretical Informatics and Applications), 24, 6, 1990.

[M93] Ron C. Mullin, A characterization of the extremal distributions of optimal normal bases, pp. 41-49, *Coding theory, Design theory*, *Group theory,* Marshall Hall Conference, Dieter Jungnickel & Scott A. Vanstone (eds.), Wiley, New York, 1993.

[MOVW89] Ron C. Mullin, I. Onyszchuk, Scott A. Vanstone & R. Wilson, Optimal normal bases in $GF(p^n)$, pp. 149-161,Discrete Applied Mathematics, 22, 1988-89.

[Mu98] Volker Müller, Efficient algorithms for multiplication on elliptic curves, Proceedings of 'GI – Arbeitskonferenz Chipkarten, TU München, 1998.

[Mu98a] Volker Müller, Fast multiplication on elliptic curves over small fields of characteristic two, pp. 219-234, Journal of Cryptology, 11, 4, 1998.

[MP97] Volker Müller & Sachar Paulus, On the generation of cryptographically strong elliptic curves, preprint, 1997.

[NS01] Phong Q. Nguyen & Igor E. Shparlinski, The insecurity of the Elliptic Curve Digital Signature Algorithm with partially known nonces, preprint, 2001.

[NR96] Kaisa Nyberg & Rainer A. Rueppel, Message recovery for signature schemes based on discrete logarithm problem, pp. 61-81, Designs, Codes and Cryptography, 7, 1996.

[O88] Eiji Okamato, Key distribution system based on identification information, pp. 194-201, LNCS 293, Advances in Cryptology – Crypto '87, Santa Barbara, California, Carl Pomerance (ed.), Springer-Verlag, 1988.

[OOSS95] Sean O'Malley, Hilarie Orman, Richard Schroeppel & Oliver Spatscheck, Fast key exchange with elliptic curve systems, pp. 43-56, LNCS 963, Advances in

Cryptology – Crypto '95, Santa Barbara, California, Don Coppersmith (ed.), Springer-Verlag, 1995.

[OW94] Paul C. van Oorschot & Michael J. Wiener, Parallel collision search with applications to hash functions and discrete logarithms, pp. 210-218, the 2nd ACM Conference on Computer and Communications Security, ACM Press, New York, November 1994.

[OW99] Paul C. van Oorschot & Michael J. Wiener, Parallel collision search with cryptanaltic applications, pp. 1-28, Journal of Cryptology, 12, 1, 1999.

[OP00] Gerardo Orlando & Christof Paar, An efficient squaring architecture for $GF(2^m)$ and its applications in cryptographic systems, pp. 1116-1117, Electronic Letters, 36, 13, June 2000.

[P93] Christof Paar, A parallel Galois field multiplier with low complexity based on composite fields, pp. 320-324, the 6th Joint Swedish-Russian Workshop on Information Theory, Mölle, Sweden, August 1993.

[P95] Christof Paar, Some remarks on efficient inversion in finite fields, p. 58 ff, 1995 IEEE International Symposium on Information Theory, Whistler, B. C., Canada, 17-22 September 1995.

[P96] Christof Paar, A new architecture for a parallel finite field multiplier with low complexity based on composite fields, pp. 856-861, IEEE Transactions on Computers, 45, 7, July 1996.

[P99] Christof Paar, Implementation options for finite field arithmetic for elliptic curve cryptosystems, the 3rd workshop on Elliptic Curve Cryptography (ECC '99), University of Waterloo, Ontario, Canada, 1999.

[PS97] Christof Paar & Pedro Soria-Rodriguez, Fast arithmetic architectures for public key algorithms over Galois field $GF((2^n)^m)$, pp. 363-378, LNCS 1233, Advances in Cryptology – Eurocrypt '97, Konstanz, Germany, Walter Fumy (ed.), Springer-Verlag, 1997.

[P99] Pascal Paillier, Public-key cryptosystems based on composite degree residuosity classes, pp. 223-228, LNCS 1592, Advances in Cryptology – Eurocrypt '99, Prague, Czech Republic, Jacques Stern (ed.), Springer-Verlag, 1999.

[P00] Pascal Paillier, Trapdooring discrete logarithms on elliptic curves over rings, pp. 573-584, LNCS 1976, Advances in Cryptology – Asiacrypt 2000, Kyoto, Japan, Tatsuaki Okamoto (ed.), Springer-Verlag, 2000.

[Pi95] Richard G. E. Pinch, Extending the Wiener attack to RSA-type cryptosystems, pp. 1736-1738, Electronics Letters, 31, 1995.

[P89] Antonio Pincin, A new algorithm for multiplication in finite fields, pp. 1045-1049, IEEE Transactions on Computers, C-38, 7, July 1989.

[P78] John M. Pollard, Monte Carlo methods for index computation (mod p), pp. 918-924, Mathematics of Computation, 32, 1978.

[QSV01] Minghua Qu, Douglas Stinson & Scott A. Vanstone, Cryptanalysis of the Sakazaki-Okamato-Mambo ID-based key distribution system over elliptic curves, 2001.

[R98] Peter de Rooij, Efficient exponentiation using precomputation and vector addition chains, pp. 389-399, LNCS 1403, Advances in Cryptology – Eurocrypt '98, Espoo, Finland, Kaisa Nyberg (ed.), Springer-Verlag, 1998.

[R97] Hans-Georg Rück, On the discrete logarithm in the divisor class group of curves, preprint, 1997.

[S90] S. Saryazdi, An extension to ElGamal public key cryptosystem with a new signature scheme, pp. 195-198, Communication, Control and Signal Processing, Elsevier, 1990.

[Sa00] Takakazu Satoh, The canonical lift of an ordinary elliptic curve over a finite field and its point counting, Journal of the Ramanujan Mathematical Society, December 2000.

[Sc93] Oliver Schirokauer, Discrete logarithms and local units, pp. 409-423, Philosophical Transactions of the Royal Society of London A, 345, 1993.

[S91] Claus-Peter Schnorr, Efficient signature generation by smart cards, pp. 161-174, Journal of Cryptology, 4, 3, 1991.

[S85] René Schoof, Elliptic curves over finite fields and the computation of square roots mod *p*, pp. 483-494, Mathematics of Computation, 44, 170, April 1985.

[S87] René Schoof, Nonsingular plane cubic curves over finite fields, pp. 183-211, Journal of Combinatorial Theory, A 46, 1987.

[S95] René Schoof, Counting points on elliptic curves over finite fields, pp. 219-254, Journal de Théorie des Nombres de Bordeaux, 7, 1995.

[S75] Arnold Schönhage, A lower bound for the length of addition chains, pp. 1-12, Theoretical Computer Science, 1, 1975.

[S77] Arnold Schönhage, Fast multiplication for polynomials over fields of charateristic 2, pp. 395-398, Acta Informatica, 7, 1977.

[Sc00] Richard Schroeppel, Elliptic curve point ambiguity resolution apparatus and method, Patent application, 2000.

[Se98] I. A. Semaev, Evaluation of discrete logarithms in a group of *p*-torsion points of an elliptic curve in characteristic *p*, pp. 353-356, Mathematics of Computation, 67, 221, January 1998.

[S98] Gadiel Seroussi, Compact representation of elliptic curve points over F_{2^n}, Research manuscript, Hewlett-Packard Laboratories, April 1998.

[Sh97] Victor Shoup, Lower bounds for discrete logarithms and related problems, pp. 256-266, LNCS 1233, Advances in Cryptology – Eurocrypt '97, Konstanz, Germany, Walter Fumy (ed.), Springer-Verlag, 1997.

[S99] Joseph H. Silverman, The xedni calculus and the elliptic curve discrete logarithm problem, Technical Report CORR 99-05, University of Waterloo, Ontario, Canada, 1999.

[SS99] Joseph H. Silverman & Joe Suzuki, Elliptic curve discrete logarithms and the index calculus, LNCS 1514, Advances in Cryptology – Asiacrypt '98, Beijing, China, Kazuo Ohta & Dingyi Pei (eds.), Springer-Verlag, 1999.

[SS97] Robert D. Silverman & J. Stapleton, 1997, preprint.

[Sm99] Nigel P. Smart, The discrete logarithm problem on elliptic curves of trace one, pp. 193-196, Journal of Cryptology, 12, 3, October 1999.

[S01] Nigel P. Smart, How secure are elliptic curves over composite extension fields?, LNCS 2045, Advances in Cryptology – Eurocrypt 2001, Innsbuck, Austria, Birgit Pfitzmann (ed.), Springer-Verlag, 2001.

[S97] Jerome Solinas, An improved algorithm for arithmetic on a family of elliptic curves, pp. 357-371, LNCS 1294, Advances in Cryptology – Crypto '97, Santa Barbara, California, Burt S. Kaliski, Jr. (ed.), Springer, 1997.

[S00] Jerome Solinas, Efficient arithmetic on Koblitz curves, pp. 195-249, Designs, Codes and Cryptography, 19, 2000.

[S67] J. Stein, Computational problems associated with Racah algebra, pp. 397-405, Journal of Computational Physics, 1, 1967.

[V96] Serge Vaudenay, Hidden collisions on DSS, pp. 83-88, LNCS 1109, Advances in Cryptology – Crypto '96, Santa Barbara, California, Neal Koblitz (ed.), Springer-Verlag, 1996.

[V85] Hugo Volger, Some results on addition/subtraction chains, pp. 155-160, Information Processing Letters, 20, 1985.

[W69] E. Waterhouse, Abelian varieties over finite fields, pp. 521-560, Ann. Sci., École Normale Supérieure, 2, 1969.

[W90] Michael J. Wiener, Cryptanalysis of short RSA secret exponents, pp. 553-558, IEEE Transactions on Information Theory, 36, 3, 1990.

[WZ98] Michael J. Wiener & Robert J. Zuccherato, Faster attacks on elliptic curve cryptosystems, pp. 190-200, LNCS 1556, Selected Areas in Cryptography – SAC '98, Springer-Verlag, 1998.

[Y91] Yacov Yacobi, Exponentiating faster with addition chains, pp. 222-229, LNCS 537, Advances in Cryptology – Crypto '90, Santa Barbara, California, Alfred J. Menezes & Scott A. Vanstone (eds.), Springer-Verlag, 1991.

[Z98] Yuliang Zheng, Shortened digital signature, signcryption and compact and unforgeable key agreement schemes, submitted to IEEE P1363a – Standard Specifications for Public-Key Cryptography: Additional techniques, 1998.

Standards

ANSI X9.62, Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ECDSA), American Bankers Association, 1999.

ANSI X9.63, Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography, American Bankers Association, 2000.

FIPS PUB 180-1 (Federal Information Processing Standards Publication): Secure Hash Standard (SHA-1), U.S. Department of Commerce – National Institute of Standards and Technology (NIST), 1995.

IEEE P1363, Standard specifications for public-key cryptography, 2000.