Simple Tutorial on Elliptic Curve Cryptography

Last updated in December 2004

Preface

For the complexity of elliptic curve theory, it is not easy to fully understand the theorems while reading the papers or books about Elliptic Curve Cryptography (ECC). But with the development of ECC and for its advantage over other cryptosystems on finite fields, more and more people express their interests in this field. This simple tutorial is just for those who want to quickly refer to the basic knowledge, especially the available cryptography schemes in this field. The whole tutorial is organised as follows. Chapter 1 introduces some preliminaries of elliptic curves. How to use elliptic curves in cryptosystems is described in Chapter 2. The final part includes some basic notions.

The whole tutorial is based on Julio Lopez and Ricardo Dahaby's work "An Overview of Elliptic Curve Cryptography" with some extensions. Many paragraphs are just lifted from the referred papers and books. Hence, I do NOT claim any right of this report. And some important subjects are still missing, including the algorithms of group operations and the recent progress on the pairing-based cryptography, etc.

Caveat. Many included schemes in this tutorial in fact cannot meet the stringent security requirement of encryption or signature and have been broken by some means in the literature. It is not recommended to use a scheme presented in the tutorial without consulting other resource for security consideration.

Late update was on December 1, 2004, with correcting many errors.

Zhaohui Cheng

m.z.cheng@mdx.ac.uk

Contents

1 Basic Conception **4**

1.1 Discrete Mathematics Fundament 4

1.1.1 Group Fundament 4

1.1.2 Field Fundament 4

1.1.3 Finite Field F_p 5

1.1.4 Finite Field F_{2^m} 5

1.2 General Elliptic Curves 7

1.3 $E(F)$ Forms an Abelian Group Under Addition Operation 7

1.4 Elliptic Curves Over F_p 8

1.5 Elliptic Curves Over F_{2^m} 8

1.6 Addition Operation on $E(F_p)$ and $E(F_{2^m})$ 9

1.7 Some Basic Concepts and Facts 9

1.7.1 Some Concepts 9

1.7.2 Some Basic Facts 9

1.8 Discrete Logarithm Problem on Elliptic Curves 10

1.9 Approaches to Look for a Curve of Nearly Prime Order 11

2 ECC in Practice **12**

2.1 ECC Domain Parameters 12

2.2 ECC System Setup 12

2.3 Key Generation 13

2.4 Public Key Validation 13

2.5 ECC Key Agreement Protocol 14

2.5.1 Static ECDH Key Agreement Protocol 14

2.5.2 ECMQV Key Agreement Protocol 15
2.6 EC Public Key Cryptosystem 15
2.6.1 Basic Method 15
2.6.2 EC-KEM 15
2.6.3 ElGamal Cryptosystem 16
2.6.4 EC ElGamal Cryptosystem 17
2.6.5 Massey-Omura Cryptosystem 17
2.6.6 Menezes-Vanstone Cryptosystem 18
2.6.7 RSA 18
2.6.8 ECRSA 18
2.7 EC Signature Protocol 19
2.7.1 DSA 19
2.7.2 ECDSA 20
2.7.3 ElGamal Digital Signature Scheme 21
2.7.4 EC ElGamal Signature Scheme 22
2.7.5 Schnorr Signature Scheme 22
2.7.6 EC Schnorr Signature Scheme 23
2.8 Point Compression 24
2.9 Message Embedding 24

Chapter 1

Basic Conception

1.1 Discrete Mathematics Fundament

1.1.1 Group Fundament

Group is an algebraic system defined on a set G with a binary operation $\diamond$ satisfying the following:

- closure: $\forall x, y \in G, x \diamond y \in G$;
- associativity: $x \diamond (y \diamond z) = (x \diamond y) \diamond z$;
- identity: $\exists e \in G, \forall x \in G : x \diamond e = e \diamond x = x$;
- inverse: $\forall x \in G, \exists y \in G : x \diamond y = y \diamond x = e$;

Abelian Group: Group satisfies *commutativity*: $\forall x, y \in G, x \diamond y = y \diamond x$.

1.1.2 Field Fundament

Field is an algebraic system defined on a set F with two binary operations $+, \times$ satisfying the following:

- $(F, +)$ is an Abelian group;
- $(F \setminus \{0\}, \times)$ is an Abelian group, where $\{0\}$ is the identity of addition and zero of multiplication;
- distributivity: $\forall x, y, z \in F : x \times (y + z) = x \times y + x \times z; (x + y) \times z = x \times z + y \times z$

• *Characteristic* of a field F, denoted by $char(F)$, is the least positive integer n satisfying $\sum_{i=1}^{n} I = 0$, where I is the identity of field multiplication.

1.1.3 Finite Field F_p

Definition: F_p for prime p, is defined on the number set $F = \{0, 1, \ldots, p-1\}$ with two operations:

- *Addition:* $\forall a, b \in F_p$, $r \equiv a + b \, mod \, p$;
- *Multiplication:* $\forall a, b \in F_p$, $r \equiv a \times b \, mod \, p$.

Quick modulo computation: Both operations need modulo operation. For special prime numbers, the modulo operation can be done quickly. For example, for a prime $p = 2^{192} - 2^{64} - 1$, if $n < p^2$, then n can be rewritten as $n = \sum_{j=0}^{5} A_j \cdot 2^{64j}$. Then $n \equiv T + S_1 + S_2 + S_3 \, mod \, p$ where

$$\begin{array}{lllll} T & = A_2 \cdot 2^{128} + & A_1 \cdot 2^{64} + & A_0 \\ S_1 & = & A_3 \cdot 2^{64} + & A_3 \\ S_2 & = A_4 \cdot 2^{128} + & A_4 \cdot 2^{64} & \\ S_3 & = A_5 \cdot 2^{128} + & A_5 \cdot 2^{64} + & A_5 \end{array}$$

This type of prime p is called a *generalized Mersene number* introduced by Solinas.

1.1.4 Finite Field F_{2^m}

The elements in F_{2^m} can be represented in three ways.

Using polynomial basis

Every element in the field F_{2^m} can be expressed as a polynomial with degree less than m ($\forall a \in F_{2^m}, \exists$ m numbers $a_i \in \{0, 1\}, a = a_{m-1}x^{m-1} + \ldots + a_1 x + a_0 = (a_{m-1} \cdots a_0)$. The operations are defined as follows:

- *Addition:* $\forall a, b \in F_{2^m}, a + b = c = (c_{m-1} \cdots c_0), c_i = a_i + b_i \, mod \, 2 = a_i \oplus b_i$
- *Multiplication:* $\forall a, b \in F_{2^m}, a \times b = c, c = (\sum_{j=0}^{m-1} a_j x^j) \times (\sum_{j=0}^{m-1} b_j x^j) \, mod \, f(x)$, where $f(x) = x^m + \sum_{j=0}^{m-1} f_j x^j$ is an irreducible polynomial with degree m.

Note: The irreducible polynomial can be trinomial $x^m + x^k + 1$ or pentanomial $x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$. Selecting the polynomial in the above forms with middle terms with minimal degree can speed the reduction modulo operation.

Using normal basis

Normal basis of F_{2^m} over F_2 is a basis of form $\{\beta, \beta^2, \beta^4, \ldots, \beta^{2^{m-1}}\}$ where $\beta \in F_{2^m}$.

Note: This normal basis always exists. The *Gaussian Normal Basis(GNB)* can speed up the multiplication operation. If $8 \nmid m$, GNB exists. If $8|m$ and the *type* of GNB T is positive, then GNB exists if and only if $p = Tm + 1$ is prime and $gcd(Tm/k, m) = 1$ where k is the multiplicative order of 2 modulo p.

Every element in F_{2^m} can be expressed as the basis linear combination, which means $\forall a \in F_{2^m}, \exists\, m$ numbers $a_i \in \{0, 1\}, a = \sum_{j=0}^{m-1} a_j \beta^{2^j} = (a_0 \cdots a_{m-1})$. The operations with GNB are defined as follow:

- *Addition:* $\forall a, b \in F_{2^m}, a + b = c = (c_0 \cdots c_{m-1}), c_i = a_i \oplus b_i$
- *Multiplication:* $\forall a, b \in F_{2^m}\ a \times b = c = (c_0 \ldots c_{m-1})$, where

$$c_l = \begin{cases} A_l & \text{if } T \text{ is even} \\ B_l & \text{if } T \text{ is odd} \end{cases}$$

  where T is the type of GNB and

$$A_l = \sum_{k=1}^{p-2} a_{F(k+1)+l} b_{F(p-k)+l}$$
$$B_l = \sum_{k=1}^{m/2} (a_{k+l-1} b_{m/2+k+l-1} + a_{m/2+k+l-l} b_{k+l-1}) + A_l$$

  with $p = Tm + 1$ and $F(1) \ldots F(p-1)$ defined by $F(2^i u^j \bmod p) = i$ for $0 \le i \le m-1, 0 \le j \le T-1$ and $u \in F_p$ has order T.

Using subfield basis

When $m = m_1 m_2$, we can regard F_{2^m} as an extension of degree m_2 of $F_{2^{m_1}}$, and we can represent the elements in F_{2^m} using the basis of the form $\{\alpha_i \beta_j : 0 \le i < m_1, 0 \le j < m_2\}$, where $\beta_0, \ldots, \beta_{m_2-1}$ form a basis of F_{2^m} over $F_{2^{m_1}}$, and $\alpha_0, \ldots, \alpha_{m_1-1}$ form a basis of $F_{2^{m_1}}$ over F_2. So the arithmetic operations can be done in two stages, with the outer section doing the operations on elements of F_{2^m} as vectors of symbols from $F_{2^{m_1}}$; and an inner section performing the operations on the symbols as binary words. Any combination of bases can be used, for example, normal basis for the outer section, and polynomial basis for the inner one.

When m_1 is large enough (4-16), m_2 is relatively small, which will benefit the operation speed. For example, if using polynomial basis, the polynomial is much shorter, and inner section operations can be finished by looking up tables.

For the inverse operation based on repeated multiplication can also be made more efficient. For any non-zero $\beta \in F_{2^m}$, write $\beta^{-1} = \frac{\beta^{s-1}}{\beta^s}$ with $s = \frac{2^m - 1}{2^{m_1} - 1}$. As β^s is in the subfield $F_{2^{m_1}}$, β^{s-1} has an optimized addition chain.

1.2 General Elliptic Curves

Elliptic Curves: Let $a_i \in F$, where F is a finite field. $\overline{F}$ is the algebraic closure of F. Let E be a cubic curve defined by (the general Weierstrass equation) $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ over F, that is $E = \{(x, y) \in \overline{F}^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\infty\}$, denoted by $E(\bar{F})$.

Note: Every cubic can be transformed into Weierstrass Equation by affine projectivity.

$$
\begin{array}{ll}
b_2 & = a_1^2 + 4a_2, \\
b_4 & = a_1a_3 + 2a_4, \\
b_6 & = a_3^2 + 4a_6, \\
b_8 & = a_1^2a_6 - a_1a_3a_4 + 4a2a6 + a_2a_3^2 - a_4^2, \\
c_4 & = b_2^2 - 24b_4, \\
c_6 & = -b_2^3 + 36b_2b_4 - 216b_6, \\
\Delta & = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6, \\
k & = 2y + a_1x + a_3, \\
4b_8 & = b_2b_6 - b_4^2, \\
1728\Delta & = c_4^3 - c_6^2, \\
j & = c_4^3/\Delta = 1728 + c_6^2/\Delta. \\
\text{weight of } x & = 2 \\
\text{weight of } y & = 3 \\
\text{weight of } a_i & = i
\end{array}
$$

Note: In the above definition, E is a set of points, whose both coordinates are in $\overline{F}$, which means E is the solution set for the equation on $\overline{F}$, but not F.

Rational Points: Let L be the algebraic extension of F ($F \subseteq L$). If both the coordinates of $P \in E$ lie in L, or $P = \infty$, we say P is L-rational. The set of L-rational points of E is denoted by $E(L)$.

1.3 $E(F)$ Forms an Abelian Group Under Addition Operation

Definition: Addition in $E(F)$
∞ is the special point to denote infinity. The addition operation is specified as follows: adding two points $u, v \in E(F)$.

1. draw a line though u and v which intersects with the curve at the third point w,
2. draw a vertical line through w which intersects with the curve at point z.

And define $u + v = z$. If $u = v \neq \infty$ then the line in step 1, is the tangent line of the curve through u. u adding ∞ means, the line drawn in step 1 is the vertical line passing

through u, because ∞ is infinitely far, and the vertical line in step 2 is the same as the line in step 1, which intersects the curve at the same point u. So $u + \infty = u$. ∞ plus ∞ is still ∞, which means ∞ is the identity of the group. The inverse of u, donated by $-u$, requires $u + (-u) = \infty$. According to the addition rule, we can find the unique $-u$ that satisfies the requirement. The formal definition of addition in $E(F)$ is as follow. Suppose $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are points on E other than ∞. If $x_1 = x_2$ and $y_1 + y_2 + a_1x_2 + a_3 = 0$, then $P + Q = \infty$. Otherwise $P + Q = (x_3, y_3)$,where

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2,$$

$$y_3 = -(\lambda + a_1)x_3 - \nu - a_3,$$

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & if P \neq Q, \\ \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & if P = Q, \end{cases}$$

$$\nu = \begin{cases} \frac{y_1x_2 - y_2x_1}{x_2 - x_1} & if P \neq Q, \\ \frac{-x_1^3 + a_4x_1 + 2a_6 - a_3y_1}{2y_1 + a_1x_1 + a_3} & if P = Q, \end{cases}$$

Note: λ is the tangent rate if $P = Q$,which can be computed from the differential coefficient $(2\partial_y y + a_1\partial_y x + a_1\partial_x y + a_3\partial_y = 3\partial_x x^2 + 2a_2\partial_x x + a_4\partial_x)$.

Torsion point: For $P \in E$, if $nP = \infty$, P is called an n-torsion point. The set of n-torsion points forms a subgroup of E, denoted by $E[n]$

1.4 Elliptic Curves Over F_p

Let $p > 3$ be a prime, $a, b \in F_p$ (field F's characteristic is > 3) satisfy that the discriminant $\triangle = 4a^3 + 27b^2 \neq 0$ ($a_1 = a_2 = a_3 = 0, a_4 = a$ and $a_6 = b$ corresponding to the general Weierstrass equation), which defines the elliptic curves without singularity. The curve is of a simple form: $y^2 = x^3 + ax + b$ with $a, b \in F_p$. Then an elliptic curve $E(F_p)$ over F_p consists of a set of points $\{P = (x, y)|y^2 = x^3 + ax + b, x, y, a, b \in F_p\}$ together with ∞ (infinity).

1.5 Elliptic Curves Over F_{2^m}

A (non-singular) elliptic curve $E(F_{2^m})$ over F_{2^m} defined by $y^2+xy = x^3+ax^2+b$ with $a, b \in F_{2^m}$ and $b \neq 0$, consists of a set of points $\{P = (x, y)|y^2 + xy = x^3 + ax^2 + b, x, y \in F_{2^m}\}$ together with ∞ (infinity). (The curve could be of another form $y^2 + a_3y = x^3 + a_4x + a_6$ with $a_3, a_4, a_6 \in F_{2^m}$ and $a_3 \neq 0$.)

1.6 Addition Operation on $E(F_p)$ and $E(F_{2^m})$

The formal definition of addition in $E(F)$ as follow:

- $P + \infty = P$,
- $P = (x, y)$'s inverse is

$$\begin{cases} -P = (x, -y) & \text{if } P \in E(F_p) \\ -P = (x, -x - y) & \text{if } P \in E(F_{2^m}) \end{cases}$$

  $P + (-P) = \infty$

- For $P(x_1, y_1) + Q(x_2, y_2) = (x_3, y_3)$,where $P \neq Q, x_1 \neq x_2$.

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, y_3 = \lambda(x_1 - x_3) - y_1 & \text{where } \lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad P, Q \in E(F_p) \\ x_3 = \lambda^2 + \lambda + x_1 + x_2 + a, y_3 = \lambda(x_1 + x_3) + x_3 + y_1 & \text{where } \lambda = \frac{y_2 + y_1}{x_2 + x_1} \quad P, Q \in E(F_{2^m}) \end{cases}$$

- For $P(x_1, y_1), y_1 \neq 0$, $2P = P + P = (x_3, y_3)$. If $y_1 = 0$, $2P = \infty$

$$\begin{cases} x_3 = \lambda^2 - 2x_1, y_3 = \lambda(x_1 - x_3) - y_1 & \text{where } \lambda = \frac{3x_1^2 + a}{2y_1} \quad P, Q \in E(F_p) \\ x_3 = \lambda^2 + \lambda + a, y_3 = \lambda(x_1 + x_3) + x_3 + y_1 & \text{where } \lambda = x_1 + \frac{x_1}{y_1} \quad P, Q \in E(F_{2^m}) \end{cases}$$

1.7 Some Basic Concepts and Facts

1.7.1 Some Concepts

• *Order* of point $P \in E(F_q)$, where $q = p$ or $q = 2^m$, is the smallest integer r such that $rP = \infty$
• *Order* of the curve, is the number of points of $E(F)$, donated by $\#E(F)$.

Note: Curve order can be computed by Schoof's algorithm or its improvements, which is needed if one selects a random curve. And normally choosing a and b to make the curve order have a large prime factor can improve the cryptography scheme's security. So this is an important parameter of the scheme to determine the system's security.

1.7.2 Some Basic Facts

- Hasse Theorem: $\#E(F_q) = q + 1 - t$, where $\mid t \mid \leq 2\sqrt{q}$
- If q is power of 2, then $\#E(F_q)$ is even. Specifically, $\#E(F_q) \equiv 0 \, mod \, 4$, if $Tr(a) = 0$; $\#E(F_q) \equiv 2 \, mod \, 4$, if $Tr(a) = 1$. $Tr(\cdot)$ is the linear map from F_{2^m} to F_2. $Tr(a) = \sum_{i=0}^{m-1} a^{2^i}$.

- $E(F_q)$ is Abelian group of rank 1 or 2, which means $E(F_q)$ is isomorphic to $Z_{n_1} \times Z_{n_2}$, where n_1 divides n_2 and $q-1$.
- If q is the power of 2, $P = (x, y) \in E(F_q)$ has odd order, then the x-coordinate of kP for all integer k has the same trace as coefficient a ($Tr(kP_x) = Tr(a)$). This property can be used to compress the presentation of points.

1.8 Discrete Logarithm Problem on Elliptic Curves

$ECDLP$: Given two points $P, Q \in E(F_q)$ on an elliptic curve, $ECDLP$ determines the integer k, satisfying $Q = kP$, where P has order n, provided that such $0 \leq k \leq n-1$ exists.

Note: Because Pohlig-Hellman algorithm reduces the computation of k to the problem of computing k modulo each prime factor of n. So if n is a large prime, the $ECDLP$ becomes harder. In practice, one must select an elliptic curve has some points (base point G) which has large prime order n, and $\#E(F_q) = n \cdot h$, where h is a small integer.

Note: To date, the most efficient general algorithm to resolve the $ECDLP$ is Pollard-ρ, which has the running time $\sqrt{\pi n}/(2r)$, where r is the parallel processor number.

Special curves have faster algorithms for $ECDLP$

- *Supersingular elliptic curves*: The MOV attack on elliptic curves shows that $ECDLP$ can be reduced to the classical discrete logarithm problem on some extension field F_{q^k}, for some integer k (k is called the embedding degree or MOV degree). The MOV attack is only practical when k is small. For supersingular elliptic curves, $k \leq 6$.

  Note: Normally, randomly selected elliptic curves have $k > log^2 q$. But to avoid the MOV attack, one should check n, the largest prime factor of curve order doesn't divide $q^k - 1$ for all small k for which the discrete logarithm problem is tractable in F_{q^k}. In practice, this checking is done for all $k, 1 \leq k \leq 30$. On the other hand, by sacrificing the speed of the group operation, we can use supersingular curves of larger q but small k to achieve the required security, for example, with a 512-bit prime q and $k = 2$. The recently developed pairing-based cryptosystems make use of the properties of supersingular curves, at the same time meet the security requirement.

- *Prime-field anomalous curves*: If $\#E(F_p) = p$, there is polynomial algorithm solving the $ECDLP$ by lifting the curve and points to Z.

- *Binary composite finite fields*: Suppose that E is an elliptic curve defined over the composite finite field F_{2^m}, where $m = r \cdot s$. The complexity of $ECDLP$ is smaller than Pollard-ρ algorithm on significant portion of elliptic curves over $F_{2^{4s}}$. But this attack doesn't apply to F_{2^m}, where m is prime.

1.9 Approaches to Look for a Curve of Nearly Prime Order

Because $\#E(F_q)$ has the great impact on the elliptic curve crytosystem's security, it is important to find approaches to determine the curve order. We can use complex multiplication to dynamically generate curves, and use Schoof's algorithm or its descendent to determine the curve order.

Chapter 2

ECC in Practice

2.1 ECC Domain Parameters

Elliptic curve parameters over the finite field F_p or F_{2^m} can be described by one septuple:

$$T = (q, FR, a, b, G, n, h)$$

- q: the prime p or 2^m that defines the field and at the same time decides the curve form;
- FR: the field representation, i.e., using which method to represent the elements in the field (polynomial basis or normal basis or subfield basis for F_{2^m}, Montgomery residue for F_p);
- a, b: the curve coefficient, depending on the security requirement;
- G: the base point, $G = (G_x, G_y)$, one element in $E(F_q)$, which has the largest order n;
- n: the order of G, large prime. $N = \#E(F_q)$ is divisible by n;
- h: $\#E(F_q)/n$.

2.2 ECC System Setup

Setting up an elliptic curve cryptosystem requires some basic choices as followed:

- An underlying field F_q, where $q = p$, $q = 2^m$ or $q = p^m$ where p is prime and greater than 3;

- A representation of the finite field elements (polynomial basis or normal basis or subfield basis for F_{2^m}, Montgomery residue for F_p);
- Algorithms implementing the operations in the field;
- An appropriate elliptic curve (selecting a and b);
- Algorithm for implementing the elliptic curve operations i.e., point addition, point scalar, inverse. (windows methods in affine or projective coordinates);
- Elliptic Curve Cryptography protocol ($ECDSA, ECDH \ldots$).

There are some other requirements on the parameters to defend some types of attack

- $\#E(F_q)$ should have a sufficiently large prime factor n to resist the parallelized Pollard ρ-attack;
- $\#E(F_q) \neq q$ to resist Semaev, Smart and Satoh-Araki attacks on anomalous curves;
- n doesn't divide $q^k - 1$ for $1 \leq k \leq 30$, to resit the MOV attack. This requirement is not inevitable. We can choose a larger q instead of large k to achieve security as well. However, a large q will slow down the speed of group operations.
- If choosing F_{2^m}, m should be prime to resist some attacks on elliptic curve based on F_{2^m} where m is composite. (subfield basis)

2.3 Key Generation

For the public key scheme, the first step is to generate the public and private key pair. *Key generation*: With the domain parameters (q, FR, a, b, G, n, h), to generate key pair, the entity does the following:

1. Select a random integer d from the interval $[1, n-1]$;
2. Compute $Q = dG$;

d is the private key, and Q is the public key.

2.4 Public Key Validation

When receiving other's public key, the entity needs to take the following steps to validate the public key's legitimacy.

1. Check that $Q \neq \infty$;

2. Check that $Q_x, Q_y \in F_q$, where Q_x and Q_y denote the x-coordinate and y-coordinate of point Q;
3. Check that Q lies on the elliptic curve defined by a and b;
4. Check that $nQ = \infty$ ($nQ = ndG = dnG = d\infty = \infty$, because G's order is n).

The public key validation without Step is called the *partial* public-key validation. Without Step 4, the entity could be attacked. However, we can carefully select h to reduce the threat.

2.5 ECC Key Agreement Protocol

2.5.1 Static ECDH Key Agreement Protocol

In the elliptic curve Diffie-Hellman key agreement scheme, entity A does the following computation:

1. Compute $P = d_A Q_B$;
2. Check that $P \neq \infty$ (very unlikely).

The shared secret is x_P. For entity B, $P = d_B Q_A = d_B d_A G = d_A d_B G = d_A Q_B$.

Note: Step 1 can also compute P as $P = hd_A Q_B$, which can resist the attacks on small subgroup.

Note: This scheme is a *key pre-distribution* (or a non-interactive key agreement). The resulting key is completely determined *a priori* by the initial keying material. The *dynamic key distribution* scheme can possibly be established as following:

1. $A \rightarrow B : r_A$;
2. $B \rightarrow A : r_B$;
3. A computes $P = r_A r_B d_A Q_B$;
4. B computes $P = r_A r_B d_B Q_A$;
5. Check that $P \neq \infty$.

Note: Obviously, the adversary can choose r_B such that $r_A r_B \equiv 1 \, mod \, n$ to force entity A to establish $P = d_A d_B G$. Instead, in step 3 (resp. step 4), entity A (resp. entity B) can compute $H(r_A \| r_B; d_A Q_B)$ (resp. $H(r_A \| r_B; d_B Q_A)$). There are many other authenticated key establishment protocols in the literature. One of the most famous protocols is ECMQV.

2.5.2 ECMQV Key Agreement Protocol

1. $A \rightarrow B : X = r_A P$;
2. $B \rightarrow A : Y = r_B P$;
3. A computes $S_A = (r_A + \tilde{X_x} d_A) \, mod \, n$ and establishes key $K = hS_A(Y + \tilde{Y_x} Q_B)$;
4. B computes $S_B = (r_B + \tilde{Y_x} d_B) \, mod \, n$ and establishes key $K = hS_B(X + \tilde{X_x} Q_A)$;

where $\tilde{X_x}$ is the first L bits of X_x with $L = [\frac{[log_2 n]+1}{2}]$, similar to $\tilde{Y_x}$.

Note: ECMQV has some desirable security attributes including key-compromise impersonation resilience and unknown key-share resilience (the unknown key-share resilience property has been demonstrated unachievable in this version of the protocol). The computation and communication overhead of the scheme are low.

2.6 EC Public Key Cryptosystem

In the following elliptic curve cryptosystems, we assume that entity A wants to send a message m to entity B securely.

2.6.1 Basic Method

• *Key generation:* Entity B selects a random integer d_B from the interval $[1, n-1]$ as private key, and publishes $Q_B = d_B G$ as the public key.

• *Encryption:*

Entity A selects a random integer k, sends ciphertext $(kG, (kQ_B)_x + m)$ to B.

• *Decryption:*

Entity B decrypts the message by computing $m + (kQ_B)_x - d_B kG_x = m + (kd_B G)_x - (d_B kG)_x = m$.

2.6.2 EC-KEM

• *Key generation:* Entity B selects a random integer d_B from the interval $[1, n-1]$ as his private key, and publishes $Q_B = d_B G$ as the public key.

• *Encryption*

1. Select a random integer r from interval $[1, n-1]$;

2. Compute $R = rG$;
3. Compute $K = hrQ_B = (K_x, K_y)$, check that $K \neq \infty$;
4. Compute $k_1 \| k_2 = KDF(K_x)$;
5. Compute $c = ENC_{k_1}(m)$;
6. Compute $t = MAC_{k_2}(c)$.

(R, c, t) is the cipertext to B.

• *Decryption*

1. Perform a partial key validation on R;
2. Compute $K = hd_BR = hrd_BG = hrQ_B$, and check that $R \neq \infty$;
3. Compute $k_1 \| k_2 = KDF(K_x)$;
4. Verify that $t = MAC_{k_2}(c)$;
5. Compute $m = ENC^{-1}_{k_1}(c)$.

Note: Random r introduces the randomness to the secret key K. h is used to resist small group attacks and R enables the entity B to recover the secret key.

Note: This is a key-encryption-message scheme. The used encryption algorithm ENC and message authentication code generator MAC could be any algorithm satisfying the security requirement, such as AES-128 and SHA-1 separately. KDF is a key derivation function implemented using a hash function. Some KEM computes $k_1 \| k_2 = KDF(R \| K_x)$ instead. The whole scheme achieves very strong security (IND-CCA2). For more details, please refer to [4]. Another advantage of this scheme is that the message embedding is not required.

2.6.3 ElGamal Cryptosystem

• *Key generation:* a prime p and a generator g of Z_p^*, the private key d_B and public key $y_B = g^{d_B} \, mod \, p$ for B.

• *Encryption*

1. Select a random integer k from the interval $[1, p-1]$, and compute $r = g^k \, mod \, p$;
2. Compute $c = my_B^k \, mod \, p$.

(r, c) is the ciphertext to B.

• *Decryption*

1. $m = cr^{-d_B} \, mod \, p$, because $cr^{-d_B} = my_B^k r^{-d_B} = mg^{d_B k} g^{-d_B k} \, mod \, p = m \, mod \, p$.

Note: The message $m \in Z_p$.

2.6.4 EC ElGamal Cryptosystem

• *Key generation:* Entity B selects a random integer d_B from the interval $[1, n-1]$ as his private key, and publishes $Q_B = d_B G$ as the public key.

• *Encryption*

1. Represent the message m as a point $M \in E(F_q)$;
2. Select a random integer r from interval $[1, n-1]$ and compute $C_1 = rG$;
3. Compute $C_2 = rQ_B + M$.

(C_1, C_2) is the ciphertext is to B.

• *Decryption*

1. $M = C_2 - d_B C_1$, because $C_2 - d_B C_1 = rQ_B + M - d_B rG = rd_B G + M - d_B rG$.
2. Recover the message m from the point M.

Note: This scheme uses $Q = rG$ to map the element in $Z_n \setminus \{0\}$ to the element in $E(F_q)$. In fact we can use other group isomorphism $f : Z_n \setminus \{0\} \to E(F_q)$ instead. We also need an injective map $I : \{0,1\}^{|m|} \to E(F_q)$ and I^{-1} to map the message m to and from a point M in the field $E(F_q)$ (the message embedding algorithm).

2.6.5 Massey-Omura Cryptosystem

• *Key generation:* This scheme only needs two entities to agree on an elliptic curve E over a finite field F_q. Entities need not to publish their public keys.

• *Encryption*

1. A represents the message m as a point $M \in E(F_q)$;
2. A chooses a secret integer r_A with $gcd(r_A, N) = 1$, computes $M_1 = r_A M$, and sends M_1 to B; $(N = \#E(F_q))$
3. B chooses a secret integer r_B with $gcd(r_B, N) = 1$, computes $M_2 = r_B M_1$, and sends M_2 to A;
4. A computes $r_A^{-1} \in Z_N$, $M_3 = r_A^{-1} M_2$, and sends M_3 to B;

• *Decryption*

1. B computes $r_B^{-1} M_3 = r_B^{-1} r_A^{-1} r_B r_A M = M$.
2. B recover the message m from the point M.

Note: The scheme needs to map the message to the point on the curve.

Note: Strictly speaking, this scheme is more like a key exchange protocol. The receiver has to be online to receive a message. Without assumption, the scheme is not secure.

2.6.6 Menezes-Vanstone Cryptosystem

• *Key generation:* Entity B selects a random integer d_B from the interval $[1, n-1]$ as the private key, and publishes $Q_B = d_B G$ as the public key.

• *Encryption*

1. Select a random integer r from interval $[1, n-1]$ and compute $C_1 = rG$;
2. Compute $c_2 = m\{rQ_B\}_x$.

(C_1, c_2) is the ciphertext to B.

• *Decryption*

1. $m = c_2/\{d_B C_1\}_x$, because $c_2/\{d_B C_1\}_x = m\{rQ_B\}_x/\{rd_B G\}_x = m\{rd_B G\}_x/\{rd_B G\}_x$.

Note: The scheme uses $Q = rG$ to map the element in $Z_n \setminus \{0\}$ to the element in $E(F_q)$. In fact we can use other group isomorphism $f : Z_n \setminus \{0\} \to E(F_q)$ instead. And this scheme doesn't need to map the message m to one point in the field $E(F_q)$.

2.6.7 RSA

• *Key generation:* two large primes p, q; $n = pq$; one prime e_B such that $e_B d_B \equiv 1 \, mod \, \phi(n)$ where $\phi(n) = (p-1)(q-1)$. B publishes e_B as the public key, and keeps d_B as the secret key.

• *Encryption*

$c \equiv m^{e_B} \, mod \, n$. c is the ciphertext of message m to B.

• *Decryption*

$c^{d_B} = m^{e_B d_B} = m \, mod \, n$.

2.6.8 ECRSA

• *Key generation:*

1. B selects two distinct primes p, q with $p \equiv q \equiv 2 \, mod \, 3$ and computes $n = pq$;
2. B chooses integers e_B, d_B with $e_B d_B \equiv 1 \, mod \, (\mathrm{lcm}(p+1, q+1))$ (he can use $(p+1)(q+1)$ in place of $\mathrm{lcm}(p+1, q+1)$). B publishes e_B as the public key, and keeps d_B as the secret key. $\mathrm{lcm}(a, b)$ is the lowest common multiple of a and b.

• *Encryption*

1. A represents the message m as a pair of integer $(m_1, m_2) \, mod \, n$ and regards it as a point M on the elliptic curve E given by $y^2 = x^3 + b \, mod \, n$, where $b = m_2^2 - m_1^3 \, mod \, n$. ($A$ doesn't need to compute b, because the encryption operation doesn't involve b.);

2. A adds M to itself e_B times on E to obtain $C = (c_1, c_2) = e_B M$

$C = (c_1, c_2)$ is the ciphertext of message m to B.

• *Decryption*

B computes $d_B C = M$ on E to obtain M.

Note: First, an element (x, y) in $E(Z_n)$ can be represented as a pair $(x_p, y_p), (x_q, y_q)$ on $E(F_p)$ and $E(F_q)$ respectively by Chinese Remainder Theorem. Second, the operation on $E(Z_n)$ is computed by the corresponding operation on $E(F_p)$ and $E(F_q)$ respectively, i.e., $d(x, y) = (d(x_p, y_p), d(x_q, y_q))$. Because of the choice of p and q, we have $\#E(F_p) = p + 1$ and $\#E(F_q) = q + 1$. So, $d_B C = d_B e_B M = (1 + k(p + 1))M = M + k(p + 1)M = M + \infty = M$ on the curve over F_p, for some $k \geq 1$. Similarly on the curve over F_q, we have $d_B C = M$. So $d_B C = M$ on the curve over Z_n.

2.7 EC Signature Protocol

In the following elliptic curve signature schemes, we assume that entity A wants to sign a message m.

2.7.1 DSA

• *Key generation*: primes q, p satisfying $q \mid (p - 1)$, the generator α of the unique cyclic subgroup of Z_p^* (satisfying $\exists u \in Z_p^*, \alpha = u^{(p-1)/q} \, mod \, p$, but $\alpha \neq 1$). A's private key d_A is a random integer from the interval $[1, q - 1]$ and A's public key is $y_A = \alpha^{d_A} \, mod \, p$.

• *Signing scheme*

1. Select a random integer k from interval $[1, q - 1]$;

2. Compute $k^{-1} \, mod \, q$;

3. Compute $r = (\alpha^k \, mod \, p) \, mod \, q$;

4. Compute $s = k^{-1}\{h(m) + d_A r\} \, mod \, q$. h is the hash function implemented using SHA-1: $\{0, 1\}^* \rightarrow F_q$.

(r, s) is A's signature of message m;

• *Verifying scheme*

1. Verify that $1 \leq r \leq q-1$ and $1 \leq s \leq q-1$;
2. Compute $h(m)$ and $w = s^{-1} \, mod \, q$;
3. Compute $u_1 = h(m) \cdot w \, mod \, q$ and $u_2 = r \cdot w \, mod \, q$;
4. Compute $v = (\alpha^{u_1} y_A^{u_2} \, mod \, p) \, mod \, q$
5. Accept if and only if $v = r$.

• *Consistency*

$h(m) = (ks - d_A r) \, mod \, q; h(m) \cdot w = (k - d_A rw) \, mod \, q; u_1 + d_A u_2 = (k - d_A rw) + d_A rw = k \, mod \, q; (\alpha^{u_1} y_A^{u_2} \, mod \, p) \, mod \, q = (\alpha^{u_1 + d_A u_2} \, mod \, p) \, mod \, q = (\alpha^k \, mod \, p) \, mod \, q$, because the way of choosing p, q and α.

Note: Some suggest using $h(r, m)$ in place of $h(m)$ and $h(r, m)$ is the hash function: $F_q^* \times \{0,1\}^* \rightarrow F_q$.

2.7.2 ECDSA

• *Key generation:* Entity A selects a random integer d_A from the interval $[1, n-1]$ as the private key, and publishes $Q_A = d_A G$ as the public key.

• *Signing scheme*

1. Select a random integer k from interval $[1, n-1]$;
2. Compute $kG = (x_1, y_1)$ and $r = x_1 \, mod \, n$. If $r = 0$ goto step 1;
3. Compute $k^{-1} \, mod \, n$;
4. Compute $e = h(m)$, where h is a hash function implemented using SHA-1: $\{0,1\}^* \rightarrow F_n$;
5. Compute $s = k^{-1}\{e + d_A r\} \, mod \, n$. If $s = 0$ goto step 1.

(r, s) is A's signature of message m.

• *Verifying scheme*

1. Verify that r and s are integers in $[1, n-1]$;
2. Compute $e = h(m)$;
3. Compute $w = s^{-1} \, mod \, n$;
4. Compute $u_1 = ew \, mod \, n$ and $u_2 = rw \, mod \, n$;

5. Compute $u_1G + u_2Q_A = (x_1, y_1)$;
6. Compute $v = x_1 mod n$;
7. Accept the signature if and only if $v = r$.

• *Consistency*

$u_1G + u_2Q_A = u_1G + u_2d_AG = (u_1 + u_2d_A)G$. If $(u_1 + u_2d_A) = k \, mod \, n$ then the scheme is correct, because n is the order of G. $e = ks - d_Ar \, mod \, n; u_1 + d_Au_2 = ew + d_Arw = es^{-1} + d_Arw = k - d_Arw + d_Arw = k \, mod \, n$.

Note: Some suggest using $h(r, m)$ in place of $h(m)$ and $h(r, m)$ is the hash function: $F_n^* \times \{0, 1\}^* \rightarrow F_n$.

2.7.3 ElGamal Digital Signature Scheme

• *Key generation*: the prime p, a generator g of field Z_p^*, A's private key d_A is a random integer from the interval $[1, p-1]$ and her public key is $y_A = g^{d_A} \, mod \, p$.

• *Signing scheme*

1. Select a random integer k from interval $[1, p-1]$, satisfying $gcd(k, p-1) = 1$;
2. Compute $k^{-1} \, mod \, (p-1)$;
3. Compute $r = g^k \, mod \, p$;
4. Compute $s = k^{-1}\{h(m) - d_Ar\} \, mod \, (p-1)$. h is the hash function:$\{0, 1\}^* \rightarrow Z_p$.

(r, s) is A's signature of message m.

• *Verifying scheme*

1. Verify that $1 \leq r \leq p-1$;
2. Compute $v_1 = y_A^r r^s \, mod \, p$;
3. Compute $h(m)$ and $v_2 = g^{h(m)}$;
4. Accept if and only if $v_1 = v_2$.

• *Verifying theorem*

$y_A^r r^s \, mod \, p = g^{d_Ar} g^{ks} \, mod \, p = g^{ks + d_Ar} \, mod \, p = g^{h(m)} \, mod \, p$, because $s = k^{-1}\{h(m) - d_Ar\} \, mod \, (p-1) \Rightarrow h(m) = ks + d_Ar \, mod \, (p-1)$. This is based on the theorem $a^j \equiv a^k \, mod \, p$, where $j \equiv k \, mod \, (p-1)$.

2.7.4 EC ElGamal Signature Scheme

• *Key generation:* Entity A selects a random integer d_A from the interval $[1, n-1]$ as her private key, and publishes $Q_A = d_A G$ as her public key.

• *Signing scheme*

1. Select random integer k from the interval $[1, n-1]$;
2. Compute $R = kG = (x_1, y_1)$, $r = x_1 \, mod \, n$; if $r = 0$ then goto step 1;
3. Compute $e = h(m)$, where h is a hash function $\{0,1\}^* \rightarrow F_n$;
4. Compute $s = k^{-1}\{e + d_A r\} \, mod \, n$. If $s = 0$ then goto step 1.

(R, s) is A's signature of message m.

• *Verifying scheme*

1. Verify that s is an integer in $[1, n-1]$ and $R = (x_1, y_1) \in E(F_q)$.
2. Compute $V_1 = sR$;
3. Compute $V_2 = h(m)G + rQ_A$, where $r = x_1$.
4. Accept if and only if $V_1 = V_2$.

• *Consistency*

$V_1 = sR = skG = \{\{h(m) + d_A r\} \, mod \, n\}G$; $V_2 = h(m)G + rQ_A = \{h(m) + rd_A\}G$. And because G's order is n, $kG = jG$ where $j \equiv k \, mod \, n$. So $V_1 = V_2$.

Note: It looks that the signing algorithm of ElGamal signature scheme is very much similar to ECDSA. However, in ECDSA, r is part of the signature, while in ElGamal signature scheme R is part of the signature and it is not possibly to recover R from r if $q \neq n$. Hence, two verifying algorithms are significantly different from each other.

Note: Some suggest using $h(r, m)$ in place of $h(m)$ and $h(r, m)$ is the hash function: $F_n^* \times \{0,1\}^* \rightarrow F_n$.

2.7.5 Schnorr Signature Scheme

• *Key generation*: primes q, p, satisfying $q \mid (p-1)$, the generator α of the unique cyclic subgroup of Z_p^* (satisfying $\exists u \in Z_p^*, \alpha = u^{(p-1)/q} \, mod \, p$,but $\alpha \neq 1$). A's private key d_A is a random integer from the interval $[1, q-1]$, and her public key is $y_A = \alpha^{d_A} \, mod \, p$.

• *Signing scheme*

1. Select a random integer k from interval $[1, q-1]$;

2. Compute $r = \alpha^k \, mod \, p$, $e = h(m\|r)$ and $s = d_A e + k \, mod \, q$. h is the hash function: $\{0,1\}^* \rightarrow Z_q$.

(s, e) is A's signature of message m.

• *Verifying scheme*

1. Compute $v = \alpha^s y_A^{-e} \, mod \, p$ and $e' = h(m\|v)$;
2. Accept if and only if $e = e'$.

• *Consistency*

$v = \alpha^s y_A^{-e} \, mod \, p = \alpha^{de+k} \alpha^{-de} = \alpha^k \, mod \, p = r$.

2.7.6 EC Schnorr Signature Scheme

• *Key generation:* Entity A selects a random integer d_A from the interval $[1, n-1]$ as her private key, and publishes $Q_A = -d_A G$ as her public key.

• *Signing scheme*

1. Select a random integer k from interval $[1, n-1]$;
2. Compute $R = kG = (x_1, y_1), r = x_1 \, mod \, n$ and $e = h(r, m)$, where $h(r, m)$ is hash function:$F_q \times \{0,1\}^{|m|} \rightarrow F_n$. If $e = 0$ then goto step 1;
3. Compute $w = k + d_A e \, mod \, n$, if $w = 0$ then goto step 1.

(e, w) is A's signature of message m.

• *Verifying scheme*

1. Verify that e, w are in the interval $[1, n-1]$
2. Compute $V = wG + eQ_A = (x_1, y_1), r' = x_1 \, mod \, n$ $(r = x_1)$ and $e = h(r', m)$;
3. Accept if and only if $e = e'$.

• *Consistency*

$V = wG + eQ_A = (k + d_A e)G + e(-d_A)G = kG = R$.

2.8 Point Compression

How to present one point $P = (x, y)$?

1. Drop y coordinate. This creates ambiguity as it is a 2:1 map
2. Drop x coordinate. This creates ambiguity as it is a 3:1 map
3. Represent P as $(x, 0)$ if $y \in \{0, 1, \ldots, \frac{q-1}{2} - 1\}$, otherwise as $(x, 1)$. Only when P is a 2-torsion point, the presentation introduces ambiguity.

Note: Using this type of presentation, the system needs to compute y every time with extra complexity.

2.9 Message Embedding

The method proposed by Koblitz represents a message as a point on an elliptic curve. Suppose E is an elliptic curve given by $y^2 = x^3 + Ax + B$ over a field F_q where q is a large prime. Using the following steps to map a message m to a point on the curve.

1. Treat m as an element in F_q and let x have the value of m.
2. Compute $\alpha = x^3 + Ax + B \, mod \, q$.
3. Find the square root β of α mod q.
   (a) Compute $\delta = \alpha^{(q-1)/2} \, mod \, q$.
   (b) If $\delta \neq 1$, set $x = x + 1$, goto Step 2.
   (c) Compute the square root β using one of the following methods.
      i. If $q \equiv 3 \, mod \, 4$, compute $u = (q - 3)/4$ and set $\beta = \alpha^u \, mod \, q$.
      ii. If $q \equiv 5 \, mod \, 8$, compute $u = (q - 5)/8$, $\gamma = (2\alpha)^u \, mod \, q$, $i = 2\alpha\gamma^2 \, mod \, q$ and set $\beta = \alpha\gamma(i - 1) \, mod \, q$.
      iii. If $q \equiv 1 \, mod \, 4$, please refer to [1].
4. If the right-most bit of β equals to $x \, mod \, 2$, then set $y = \beta$. Otherwise, set $y = q - \beta$.
5. Output the point (x, y).

Note: For other curves, please refer to [1] for details.

Some Notions

- The **projective n-space** over a field K, denoted by $P^n(K)$, is the set of equivalence classes $(K^{n+1}/\{0,\ldots,0\})/\sim$, where $(x_0,\ldots,x_n) \sim (y_0,\ldots,y_n)$ if and only if $\exists\lambda \in K^*$ such that $x_i = \lambda y_i$ for $0 \leq i \leq n$.

- **Homogenous polynomial:** Polynomials whose sum of powers of variables in each item is the same.

- The **Weierstrass equation in homogenous form** is the equation of degree 3 of the form: $Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^6$, where $a_1, a_2, a_3, a_4, a_6 \in K$.

- **Singularity:** The elliptic curve in homogenous equation is said to be **non-singular** if and only if for all the projective points $P = (X : Y : Z) \in P^2(\overline{K})$ satisfying $F(X,Y,Z) = 0$, at least one of the partial derivatives: $\frac{\partial_F}{\partial_X}, \frac{\partial_F}{\partial_Y}, \frac{\partial_F}{\partial_Z}$ is not zero, where $F(X,Y,Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^6$. If all the partial derivatives are zero at the point P, P is called a **singular point**, and the curve is said to be singular.

- The **trace** of Frobenius t at q is defined by $tr(E) = t = q + 1 - \#E(F_q)$, where the elliptic curve E is over F_q.

- **Supersingularity:** An elliptic curve over F_q is **supersingular** if the trace of $E, tr(E) = q + 1 - \#E(F_q)$, is divisible by the characteristic p of F_q. If $tr(E) = 0$ when $p > 5$, the curve is supersingular.

- **Prime-field-anomalous curve:** An elliptic curve over F_p is said to be prime-field-anomalous if $\#E(F_p) = p$ i.e., $tr(E) = 1$.

- **Binary anomalous curve:** Elliptic curves on F_{2^m}, with $b = 1$ and $a = 1$ or 0 ($y^2 + xy = x^3 + 1$, or $y^2 + xy = x^3 + x^2 + 1$). $(tr(E) = 1)$

Bibliography

[1] ANSI. "Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", American National Standard, X9.62-1998.

[2] Julio Lopez and Ricardo Dahaby. "An Overview of Elliptic Curve Cryptography", *Report*, May 2000.

[3] Kiyomichi Araki, Takakazu Satoh and Shinji Miura. "Overview of Elliptic Curve Cryptography", Lecture Notes in Computer Science Vol.1431, 1998.

[4] ISO. "Encryption algorithms Part 2: Asymmetric ciphers", CD 18033-2, 2004.

[5] A. Menezes, P. van Oorschot and S. Vanstone. "Handbook of Applied Cryptography", CRC Press, 1996.

[6] Lawrence C. Washington. "Elliptic Curves Number Theory and Cryptography", CRC Press, 2003

[7] Ian Blake, Gadiel Seroussi and Nigel Smart. "Elliptic Curves in Cryptography", Cambridge University Press, 1999