Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 5 (4): 604-633, 2005
ISSN 1812-5654

Theory and Implementation of Elliptic Curve Cryptography

Kefa Rabah
Department of Physics, Eastern Mediterranean University, Gazimagusa, North Cyprus, via Mersin 10, Turkey

Abstract: This work describes the mathematics needed to implement Elliptic Curve Cryptography (ECC) with special attention to its implementation in Galois Field. Here the functionality of ECC, its advantages and challenges over other cryptosystems are also explained. Comparison with other cryptographic systems will also be undertaken based on aspects such as efficiency, size of the key needed to attain a certain level of security (this has implications on computational costs and time), known and probable attacks, current and predicted future attacks (based on the current growth of technology) and their prevention techniques. ECC reliability will also be looked into.

Key words: Elliptic Curve Cryptography, internet security and attacks, wireless, Secure Socket Layer

INTRODUCTION

With the rapid growth of the use of computers to exchange information electronically, the physical way of providing security by locks, sealing and signing documents and so on, is eliminated. However the need to exchange information securely is still very important and is therefore provided in electronic documents; usually by encryption and digital signatures. The science of keeping messages secure is called cryptography. Cryptography involves encryption and decryption of messages. Encryption is the process of converting a plaintext into ciphertext by using an algorithm and decryption is the process of getting back the encrypted message (Fig. 1). A cryptographic algorithm is the mathematical function used for encryption and decryption. In addition to providing confidentiality, cryptography is often required to provide Authentication, Integrity and Non-repudiation.

The essence of cryptography is traditionally captured in the following problem: Two parties (the tradition is to call them Bob and Alice) wish to communicate over an insecure public communication channel in the presence of malevolent eavesdropper (the tradition Eve). Bob and Alice could be military jets, e-business or just friend trying to have a private conversation (Fig. 1). They can't stop Eve listening to their radio signals (or tapping their phone line, or whatever), so what can they do to keep their communication secret? One solution is for Alice and Bob to exchange a digital key, so they both know it, but it's otherwise secret.

Moreover, it is worthy to note right from the start that the hardest part of computer security is the piece between the computer and the user. While the hardest part of encryption is maintaining the security of the data when it's being entered into the keyboard and when it's being displayed on the screen. In case of the digital signatures, the hardest part is proving that the text signed is the same text that the user viewed. And finally the hardest part of computer forensics is to know who is sitting in front of a particular computer at any time.

There are two popular kinds of cryptographic protocols: symmetric-key and asymmetric-key protocols. In the symmetric-key protocols, a common key (the secret-key) is used by both communicating partners to encrypt and decrypt messages[1]. Among these are DES, IDEA and AES. These symmetric-key cryptosystems provide high speed key and communication but have the drawback that a common (or session) key must be established for each pair of participants[2]. However, in 1976, Diffie and Hellman[3] introduced Public-Key Cryptography. The encoding function here is a trapdoor function-one whose inverse is impractical to implement, unless some extra information is available. This extra information (called the decrypting-key) is not required for encrypting the message, yet is essential for decrypting it in reasonable time. This makes it much easier to encrypt messages than to decrypt them. The beauty of such a system is that the encrypting process need not be kept secret. Each user has his own or a personal encrypting-function, which is public information (hence the name Public-Key) and a decoding key, which he keeps secret.

Corresponding Author: Kefa Rabah, Department of Physics, Eastern Mediterranean University, Gazimagusa, North Cyprus, via Mersin 10, Turkey E-mail: kefa.rabah@emu.edu.tr

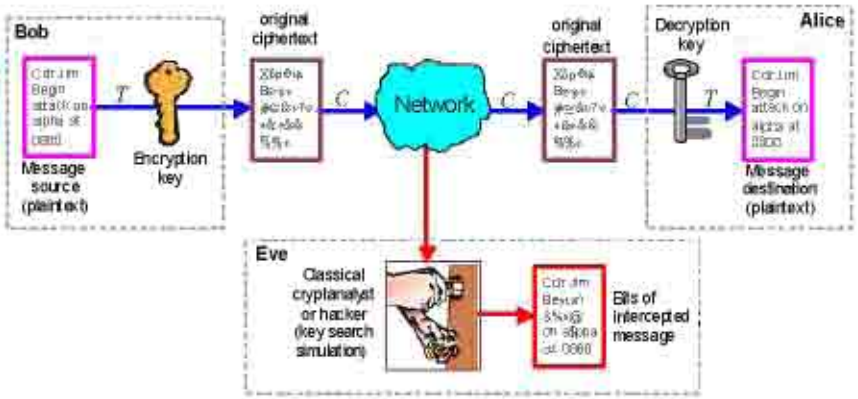

Fig. 1: Shows a schematic classical cryptographic data communication

The public-key protocol employs a pair of different but associated keys. One of these keys (the public-key) is used either for encryption (signature) of the messages and; a different key (the private-key) is used for either decryption (confidentiality) of the message[4]. Different public-key cryptographic systems are used to provide public-key security. Among these we can mention the RSA[5], Diffie and Hellman[3] (DH) key exchange algorithm, digital signature algorithm (DSA)[6,7] and ElGamal[8] cryptosystem. These systems provide these services by relying on the difficulty of different classical mathematical problems, hence provide the services in different ways.

The public-key cryptosystems are, however, slower than the symmetric ones but provide arbitrary high levels of security and do not require an initial private key exchange between two communicating parties. In the asymmetric protocol the public-key is released to the public while the other, the private-key, is known only to its owner. Because of this feature, these cryptosystems are considered to be indispensable for secure communication and authentication over open (insecure) networks[9]. It is designed to be computationally intractable to calculate a private-key from its associated public-key; that is, it is believed that any attempt to compute it will fail even when up-to-date technology and equipment are used[10]. With a public-key cryptosystem, the sender can encrypt a message using the receiver's public key-without needing to know the private-key of the receiver. Therefore, they are suitable for communication among the general public. Public-key cryptosystems can also be used to make a digital signature[11].

However, in real applications, both symmetric and asymmetric protocols are used. The public-key algorithm is first used for establishing a common symmetric-key over insecure channel. Then the symmetric system is used for secure communication with high throughput. Due to comparative slowness of the public-key algorithms, dedicated hardware is desirable for efficient implementation and operation of the cryptographic systems.

In the mid 1980s researchers noticed that another source of hard problems might be discovered by looking at the elliptic curves[12,13]. The invention of Elliptic Curve Cryptography (ECC) offered a new level of security for public key cryptosystems[14-16], which provide both encryption and digital signatures services. One potential use of elliptic curves is in the definition of public-key cryptosystems that are close analogs of existing schemes like RSA, ElGamal, DSA and DH etc. Furthermore, elliptic curve can provide versions of public-key methods that, in some cases, are faster and use smaller keys, while providing an equivalent level of security. Their advantage comes from using different kind of mathematical group for public-key arithmetic.

To date many research papers in Elliptic Curve Cryptography (ECC) have been published by researchers all over the world, as can be viewed in the refs. 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.

Introduction and history of elliptic curve: The name "elliptic curve" is based on the ellipse. Elliptic curves were first discovered after the 17th century in the form of Diophantine equation[17], $y^3-x^3 = c$, for $c \in \square$. Further, it is important to note that, however, it is easy to calculate the surface of the ellipse, it is hard to calculate the circumference of the ellipse. The calculation can be reduced to an integral:

$$\int \frac{1}{\sqrt{x^3 + Ax + B}} dx \qquad (1)$$

This integral, which cannot be solved easily, was the reason to consider the curve $Y^2 = X^3+AX+B$. This was done already during the 18th century. Probably, it was Abel in ca. 1820 who introduced the addition of points on the curve. At the moment there are several definitions for an elliptic curve. However, the following definition is used usually:

Definition: An elliptic curves E, defined over an arbitrary field K, is a non-supersingular plain projective third degree curve over K with a K-rational point O (i.e., with coordinates in K) over curve E.

Such a curve can be described by its so-called Weierstraβ form, in homogeneous coordinates x,y,z:

$$E: y^2z+a_1xyz+a_3yz^2 = x^3+a_2x^2z+a_4xz^2+a_6z^3 \quad (2)$$

where, a_1 ,...., $a_6 \in K$, such that the discriminant $\Delta \neq 0$. This discriminant is a polynomial expression in the coefficient a_1 ,...., a_6. The restriction $\Delta \neq 0$ is necessary and sufficient for E in order to be a non-singular. The curve E has exactly one K-rational point at 'infinity', i.e., z = 0, the point (0 : 1 : 0). This point plays the role of O (origin). Sometimes we want to express that a curve E is based over field K. We do this by notation E/K.

In general, it will restricted ourselves only to the affine part $z \neq 0$ of elliptic curves E:

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

For fields *K* of characteristic >2 such as ¤, ¡, £ or F_p with $p>3$ we can transform the Weierstraβ form for the affine curve by the coordinate transform:

$$\chi = x + \frac{a_1^2 + 4a_2}{12} \quad \text{and} \quad y = y + \frac{a_1}{2}x + \frac{a_3}{2} \quad (4)$$

to a curve of the form: E/K: $Y^2 = X^2+aX+b$

where, $a,b \in K$ such that the discriminant $\Delta = 4a^3+27b^2 \neq 0$. This form also known as Weierstraβ short form will be used later.

Remark: For practical applications finite field of the form $K = GF(2^m) = F_{2^m}$ are very important. For such elliptic curves the theory as mentioned above has to be modified.

In 1955, Yutaka Taniyama asked some questions about elliptic curves, i.e., curves of the form $y^2 = x^3+ax+b$ for constants a and b[18]. Hellegouarch[19] studied the application of elliptic curves for solving Fermat's Last Theorem in 1971. Elliptic curves can also be looked at as mathematical constructions from number theory and algebraic geometry, which in recent years have found numerous applications in cryptography[12].

Elliptic curve cryptosystem (ECC) is relatively new. The ECC was first introduced by Miller[13] and independently by Koblitz[12] in the mid 1980s and today it has evolved into a mature public-key cryptosystem. It was also recently endorsed by the U.S. government[20]. The ECC from the very beginning was proposed as an alternative to established public-key systems such as DH, DSA, RSA and ElGamal cryptosystems. This is so, because elliptic curves do not introduce new cryptographic algorithms, but they implement existing public-key algorithms using elliptic curves. In this way, variants of existing schemes can be devised that rely for their security on a different underlying hard problem.

Elliptic curve scheme: An elliptic curve can be defined over any field (e.g., real, rational, complex). However, elliptic curves used in cryptography are mainly defined over finite fields[21]. An elliptic curve consists of elements (x, y) satisfying the equation $y^2 = x^3+ax+b$ together with a single element denoted by O, called the "point at infinity," which can be visualized as the point at the top and bottom of every vertical line. Addition of two points on an elliptic curve is defined according to a set of simple rules (e.g., point P_1 plus point P_2 is equal to point $-P_3$ as shown in Fig. 2). The addition operation in an elliptic curve is the counterpart to modular multiplication in common public-key cryptosystems and multiple addition is the counterpart to modular exponentiation[22,23], as will be seen later.

Elliptic curve systems base their difficulty on the elliptic curve version of the DLP, which is simply called the Elliptic Curve Discrete Logarithm Problem (ECDLP). Here, the underlying field of integers modulo prime p is replaced by points on an elliptic curve defined over a finite field. Since the ECDLP is significantly harder than the DLP, even a sophisticated hacker would require most of the world's computing power for a few years to break

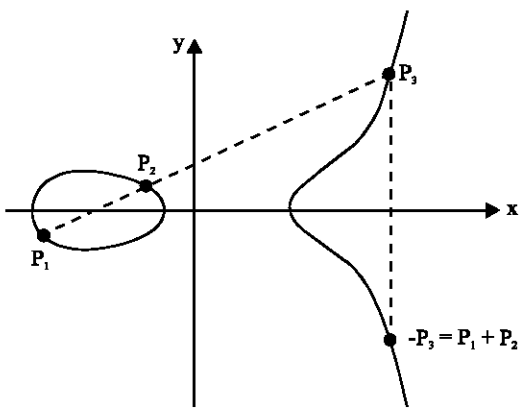

Fig. 2: Elliptic curve

an elliptic curve cryptosystem. The implementation of elliptic curve cryptography requires several choices like the type of finite field, algorithm for implementing the elliptic group operation and elliptic curve protocols which influence the performance of ECC. Elliptic curves have further shown themselves to be remarkably useful in a range of applications including primality testing and integer factorization[24-27].

Elliptic Curve Cryptosystems (ECCs) also include key distribution, encryption and digital signature algorithms. The key distribution algorithm is used to share a secret-key, the encryption algorithm enables confidential communication and the digital signature algorithm is used to authenticate the signer and validate the integrity of the message[28,29].

The mechanics of finite field F_p: Abstractly a finite field consists of a finite set of objects called field elements F together with the description of two operations-addition and multiplication-that can be performed on pairs of field elements. These operations must possess certain properties. It turns out that there is a finite field containing q field elements if and only if q is a power of a prime number and furthermore that in fact for each such q there is precisely one finite field. The finite field containing q elements is denoted by F_q. Here only two types of finite fields F_p are used-finite fields F_p with q = p, p is an odd prime which are called prime finite fields and finite fields F_{2^m} with $q = 2^m$ for some m under binary operation. The order of a finite field is the number of elements in the field. There exists a finite field of order q if and only if q is a prime power, then there are, however, many efficient implementations of the field arithmetic in hardware or in software[12,15,21,30,31].

If in adding the multiplicative identity 1 to itself in F never gives 0, then we say that F has characteristic zero; in that case F contains a copy of the field of rotational numbers. Otherwise, there is a prime number p such that 1+1+...+1 (p times) equals 0 and p is called the characteristics of the field. In that case F contains a copy of the field Z/pZ, which is called its prime field[32].

If $q = p^m$ where p is a prime and m is a positive integer, then p is called the characteristic of F_p and m is called the extension degree of F_p. Most standards which specify the elliptic curve cryptographic technique restrict the order of the underlying finite field to be odd prime (q = p) or a power of 2 ($q = 2^m$). This study describe the elements, operations and implementation of the finite field F_p, while the elements and operations of F_{2^m} can be found elsewhere[33,34].

Let p be a prime number. The finite field F_p called a prime field, is comprised of the set of integers {0,1,2,...., p-1} with the following arithmetic operations[22]:

- Addition: If $a, b \in F_p$, then a+b =r, where, r is the remainder when a+b is divided by p and $0 \le r \le p-1$. This is known as addition modulo p.
- Multiplication: If $a, b \in F_p$, $a \cdot b = s$ where, s is the remainder when, $a \cdot b$, is divided by p and $0 \le s \le p-1$. This is known as multiplication modulo p.
- Inversion: If a is a non-zero element in F_p, the inverse of a modulo p, denoted a^{-1}, is the unique integer $c \in F_p$ for which $a \cdot c = 1$.

The mechanics of elliptic curves: An elliptic curve over F_q is defined in terms of the solutions to an equation in F_q. The form of the equation defining an elliptic curve over F_q differs depending on whether the field is a prime finite field or a characteristic 2 finite field.

Roughly speaking, an elliptic curve is the solution set of a cubic equation in two variables. For number-theoretic purposes, the equation can be brought into the short Weierstraβ form, i.e., E: $y^2 = x^3+ax+b$, with integer coefficients a and b, although other forms are often used as well. The solution set is relative to some field of definition, such as the field of complex numbers. Of greatest interest are the rational solutions of such equations. The rational points on the elliptic curve E are the points over $E_p(a, b)$ that satisfy the defining equation. If the set of parameters (a, b, p) are specified, the number of rational points on the elliptic curve is determined uniquely; this number is called the order of the elliptic curve E and is denoted by #E. It is known that rational points form an additive group in the addition over the elliptic curve shown in Fig. 2. But much of the theory-and essentially all the cryptographic applications-lie in the solutions mod p (or, more generally, in solutions over finite fields). One of the nice features of elliptic curves mod p is that the size of the solution set is never too far from p. The exact theorem, proved by Hasse[35] in 1933, is $|N_p-p| \le 2p^{1/2}$ where, N_p is the number of solutions mod p. This ensures that for large p, there are lots of solutions over the finite field F_p.

But the nicest feature of elliptic curves, over any field, is that the solution set, with an extra "point at infinity" tacked on, forms a group, with a group law given

by an explicit pair of rational functions. The group turns out to be abelian (hence the additive notation) and the point at infinity, usually denoted by O, is its "zero" element. In many cases, the order of the group (over F_p) is itself a large prime, q, or a small multiple of such a prime. When that happens, the curve is ripe for use in cryptography. In particular, it can be used to attach a digital signature to a message.

Elliptic curves over galois fields: The core of the ECC is when it is used with Galois Field it becomes a one way function i.e., the math's needed to compute the inverse is not known. The main reason for attractiveness of ECC is the fact that there is no sub-exponential algorithm known to solve the discrete logarithm problem on properly chosen elliptic curve. This means that significantly smaller parameters can be used in ECC than in other competitive systems such as RSA, DH and DSA. This helps in having smaller key sizes and hence, faster computations[36].

Let an elliptic curve group over the Galois Field $E_p(a, b)$ where, p>3 and be prime, is the set of solutions or points P = (x, y) such that (x, y ϵ $E_p(a, b)$) that satisfy the equation:

$$y^2 = x^3+ax+b(\text{mod } p) \quad (6)$$

for $0 \le x < p$ together with the extra point O called the point at infinity. For a given point $P = (x_p, y_p)$, x_p and y_p are called the x and y coordinates of P, respectively. The number of points on $E_p(a, b)$ is denoted by $\#E(F_p)$. The Hasse Theorem states that[35]:

$$p + 1 - 2\sqrt{p} \le \#E(F_p) \le p + 1 + 2\sqrt{p} \quad (7)$$

The constants a and b are non negative integers smaller than the prime number p and must satisfy the condition:

$$4a^3+27b^2 \neq 0(\text{mod } p) \quad (8)$$

For each value of x, one needs to determine whether or not it is a quadratic residue. If it is the case, then there are two values in the elliptic group. If not, then the point is not in the elliptic group $E_p(a,b)$.

We should first explain why we have required the coefficients of the cubic polynomial in Eq. (6) to satisfy, $4a^3+27b^2 \neq 0 \bmod p$. Notice that:

$$\Delta = \left(\frac{a}{3}\right)^3 + \left(\frac{b}{2}\right)^2 = \frac{4a^3 + 27b^2}{4 \times 27} \quad (9)$$

is the discriminant of the cubic polynomial $f(x) = x^3+ax+b$. If $\Delta = 0$ then $f(x) = 0$ has at least a double zero X (root which makes f(x) = 0) and clearly (x, 0) is on E. For $F(x, y) = y^2-x^3-ax-b = 0$, this point satisfy:

$$\frac{\partial F}{\partial y} = 2y|_{y=0} = \frac{\partial F}{\partial x}|_{x=x} = 0 \quad (10)$$

That is, (X, 0) is a singular point at which there is no definition for a real tangent value. With the tangent-and-chord operation failing at the singular point (X, 0), E cannot be a group.

Elliptic curves over a characteristic 2 finite field $GF(2^m)$ which has 2^m elements have also been constructed and are being standardized for use in ECCs as alternative to elliptic curves over prime field finite field[24].

Construction of an elliptic curve over F_p: Let the prime number p = 23 and consider an elliptic curve E: $y^2 = x^2+x+4 \bmod 23$ defined over F_{23}. From Eq. (6) and let the constants a = 1 and b = 4. It is verified that:

$$4a^3+27b^2 \bmod p = 4(1)^3+27(4)^2 \bmod 23$$
$$= 436 \bmod 23 = 22 \neq 0 \quad (11)$$

Therefore, E is indeed an elliptic curve. We then determine the quadratic residues Q_{23} from the reduced set of residue $Z_{23} = \{1,2,3,...,21,22\}$:

Table 1: Quadratic residues of Q_{23}

$x^2 \bmod p$	$(p-x)^2 \bmod p$	=
$1^2 \bmod 23$	$22^2 \bmod 23$	1
$2^2 \bmod 23$	$21^2 \bmod 23$	4
$3^2 \bmod 23$	$20^2 \bmod 23$	9
$4^2 \bmod 23$	$19^2 \bmod 23$	16
$5^2 \bmod 23$	$18^2 \bmod 23$	2
$6^2 \bmod 23$	$17^2 \bmod 23$	13
$7^2 \bmod 23$	$16^2 \bmod 23$	3
$8^2 \bmod 23$	$15^2 \bmod 23$	18
$9^2 \bmod 23$	$14^2 \bmod 23$	12
$10^2 \bmod 23$	$13^2 \bmod 23$	8
$11^2 \bmod 23$	$12^2 \bmod 23$	6

Therefore, the set of (p-1)/2 = 11 quadratic residues $Q_{23} = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$. Now, for $0 \le x < p$, compute, $y^2 = x^3+x+4 \bmod 23$ and determine if y^2 is in the set of quadratic residues Q_{23}:

Table 2: Quadratic residues of Q_{23} and their roots

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
y^2	4	6	14	11	3	19	19	9	18	6	2	12	19	6	2	13	22	12	12	5	20	17	2
$y^2 \in Q_{23}$?	yes	yes	no	no	yes	no	no	yes	yes	yes	yes	yes	no	yes	yes	yes	no	yes	yes	no	no	no	yes
y_1	2	11			7			3	8	11	5	9		11	5	6		9	9				5
y_2	21	12			16			20	15	12	18	14		12	18	17		14	14				18

Hence, the points in $E(F_{23})$ are O and the following:

$$E_{23}(1,4) = \begin{Bmatrix} (0,2) & (0,21) & (1,11) & (1,12) & (4,7) & (4,16) & (7,3) \\ (7,20) & (8,8) & (8,15) & (9,11) & (9,12) & (10,5) & (10,18) \\ (11,9) & (11,14) & (13,11) & (13,12) & (14,5) & (14,18) & (15,6) \\ (15,17) & (17,9) & (17,14) & (18,9) & (18,14) & (22,5) & (22,18) \end{Bmatrix} \quad (12)$$

Figure 3 shows a scatterplot of the elliptic group $E_p(a, b) = E_{23}(1, 4)$.

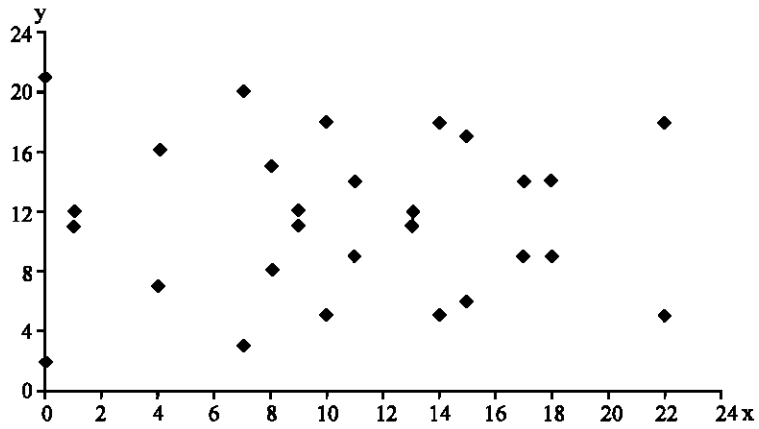

Fig. 3: Scattaerplot of elliptic group E_p (a, b) = E_{23}(1, 4)

In real practice p is chosen to be a large prime number. Take, for example, a large group of points with prime number:

p = 6, 227,101,735,386,680,763,835,789,423,207,666,416,083,908,700,390,324,961,279

Next imagine a square grid (Fig. 3) with this many rows and columns. There is a curve defined over this space of the form: E: $y^2 = x^3+ax+b \pmod p$ where, a and b are two more carefully chosen large numbers so that the curve is not weak and $4a^3+27b^2 \neq 0 \pmod p$. This curve contains exactly N points, where:

N = 6,277,101,735,386,680,763,835,789,423,337,720,473,986,773,608.255,189,015,329

These points form a group, according to the rule above, which is ideal for elliptic curve Diffie-Hellman (ECDH) algorithm. Modern computers have no trouble dealing with numbers of this size, which are actually much smaller than those used in traditional DH and RSA cryptosystems. If you look at p as binary number you'll see that it has a very special form, $p = 2^{192}-2^{64}-1$, which makes computation much easier. It is interesting to note that p and N are so "close" to each other, relatively speaking; they only differ in the lower half of their bits. Elliptic curve theory predicts this.

Addition and multiplication operation over elliptic groups: There is a rule called the chord-and-tangent rule, for adding two points on an elliptic curve $E(F_p)$ to give a third elliptic curve point as was discussed earlier in Section 2.0 Together with this addition operation, the set points $E(F_p)$ forms with O serving as the identity. It is this group that is used in the construction of elliptic curve cryptosystems. The addition rule is best explained geometrically. Let $P = (x_1, x_1)$ and $Q = (x_2, y_2)$ be two distinct points on an elliptic curve E (Fig. 4). Then the sum of P and Q, denoted by $R = (x_3, y_3)$, is defined as follows:

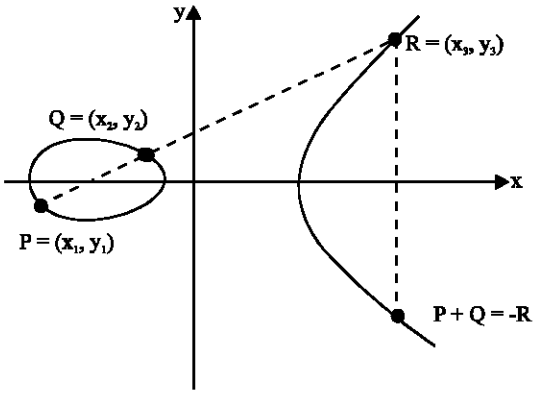

Fig. 4: Geometric description of the addition of two ditinct elliptic curve P+Q = R

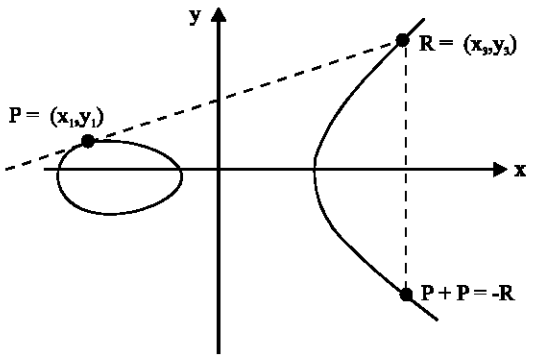

Fig. 5: Geometric description of the addition of two distinct elliptic curve P+P = R

First draw the line through P and Q; this line intersects the elliptic curve in a third point. Then R is the reflection of this point in the x-axis. The elliptic curve in the Fig. 4 consists of two parts, the ellipse-like figure and the infinite curve.

If $P = (x_1, x_1)$, then the double of P, denoted by $R = (x_3, y_3)$, is defined as follows: First draw the tangent line to the elliptic curve at P. This line intersects the elliptic curve in a second point. Then R is the reflection of this point in the x-axis (Fig. 5).

The following algebraic formulae for the sum of two points and double of a point can now be derived from the geometric description in the next two sections.

The mechanics of elliptic curve arithmetic operations: All of the arithmetic operations make perfectly good sense modulo p, provided that the denominators are relatively prime to p. Specifically, we define the operation ∂ modulo p (O is the identity) by the following arithmetic operation:

Let the points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be in the elliptic group $E_p(a, b)$ and O be the point at infinity. The rules for addition over the elliptic group $E_p(a, b)$ are:

1. $P+O = O+P$ for all $P \in E(F_p)$
2. If $P = (x, y) \in E(F_p)$ and if $x_2 = x_1$ and $y_2 = -y_1$, that is $P = (x_1, x_2)$ and $Q = (x_1, x_2)$, $(x_1, -y_1) = -P$ then $P+Q = O$ (observe that -P is indeed a point on the curve).
3. Point addition: Let $P = (x_1, y_1) \in E(F_p)$ and $Q = (x_2, y_2) \in E(F_p)$, If $P \neq \pm Q$, then their sum $P+Q = (x_3+y_3)$.
4. Point doubling: Let Let $P = (x_1, y_1) \in E(F_p)$ and, If $P \neq -P$. Then $2P = (x_3+y_3)$.

Define x_3 and y_3 by:

$$x_3 = (\lambda^2 - x_1 - x_2) \bmod p$$
$$\text{and} \qquad y_3 = \lambda(x_1 - x_3) - y_1 \bmod p \qquad (13)$$

where:

$$\lambda = \begin{cases} y_2 - y_1 / x_2 - x_1 & \text{if } P \neq Q, \text{Point addition} \\ 3x_1^2 + a/2y_1 & \text{if } P = Q, \text{Point doubling} \end{cases} \qquad (14)$$

The properties of binary operation

1. Given an elliptic curve and two rational points on that curve: (x_1, y_1) and $(x_2, y_2) \neq (x_1, -y_1)$, we define a binary operation:

$$(x_1, y_1) \partial (x_2, y_2) = (x_3, -y_3) \qquad (15)$$

where, x_3 and y_3 are defined by Eq. (13). Note that the sum of the two points is not the third point on that line, but the reflection across the x-axis of that third point as shown in Fig. 4 and 5. It is still on the same elliptic curve.

2. We now define a set, namely the rational points on an elliptic and a binary operation. We would like to make this into a group. To do this, we need to define a binary operation $(x, y)\partial(x, -y)$, we also need to find an identity and we have to find inverses. We can solve all our problems with one stroke as will be seen below.
3. Next we define O to be the identity for the binary operation ∂ and define:

$$(x, y)\partial(x, -y) = (x, -y)\partial(x, y) = O \qquad (16)$$

The point O can be thought of as a point infinitely far north so that every vertical line passes through it. One of the beauties of this definition is that now every straight line which intersects the curve at two points also intersects at a third (Fig. 4 and 5).

Then the binary operation, ∂ mod p, is then given by:

$$(x_1, y_1)\partial(x_2, y_2) = (x_3, y_3) \bmod p \quad (17)$$

where, x_3 and y_3 are defined. In particular, if p is an odd prime then our binary operation is always defined.

Observe that the addition of two elliptic curve points in $E(P_f)$ requires a few arithmetic operations (addition, subtraction, multiplication and inversion) in the underlying field F_p. We should also recall that gcd(a, b) = 1, then there exist an inverse of b such that: $a \times b = 1 \bmod n$, that is b is inverse of modulo n.

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

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

Implementation of multiplication/addition over an elliptic curve group modulo p: The multiplication over an elliptic curve group E_p (a, b) is the equivalent operation of the modular exponentiation in RSA.
Let P = (7, 3) $\in E_{23}$ (1, 4). Then 2P = (x_3, y_3) is equal to: 2P = P+P = $(x_1, y_1)+(x_1, y_1)$
If (x_1, y_1) = (7, 3), then we can determine the values of λ, x_3 and y_3 by using Eqs. (13) and (14) as follows:

$$\lambda = \frac{3x_1^2 + a}{2y_1} \bmod p = \frac{3(7)^2 + 1}{2(3)} \bmod 23 = \frac{148}{6} \bmod 23 = \frac{5}{3} \bmod 23 = 17 \bmod 23$$

$$x_3 = \lambda^2 - x_1 - x_2 \bmod p = (17)^2 - 7 - 7 \bmod 23 = 275 \bmod 23 = 22$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \bmod p = 17(7 - 22) - 3 \bmod 23 = 18$$

Therefore, 2P = (x_3, y_3) = (22, 18).
Next, let P = (4, 16) $\in E_{23}$(1, 4) and Q = (14, 18) $\in E_{23}$(1, 4). ThenP+Q = (x_3, y_3) is given by:

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod 23 = \frac{18 - 16}{14 - 4} \bmod 23 = \frac{1}{5} \bmod 23 = 14 \qquad (\text{since } 5\lambda = 1 \bmod 23)$$

$$x_3 = (\lambda^2 - x_1 - x_2) \bmod 23 = (14^2 - 4 - 14) \bmod 23 = 17$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \bmod 23 = 14(4 - 17) - 16 \bmod 23 = -14 \bmod 23 = 9$$

Hence P+Q = (17, 9). The set of points on $E(F_p)$ forms a group under this addition rule. Furthermore, the group is abelian-meaning that $P_1+P_2 = P_2+P_1$ for all points $P_1, P_2 \in E(F_p)$. The neutral point is the point at infinity.

Scalar point multiplication: basic methods: One crucial operation is scalar point multiplication since it determines the speed of an elliptic curve cryptosystem. That is, given an integer k and a point $P. \in E(F_p)$, a scalar multiplication is the process of adding P to itself k times. By definition, $\underbrace{kP = P_{14} + P_{42} + K_4 + P_{63}}_{k \text{ times}}$. This problem is analogous to raising an element to the k-th power in the multiplicative subgroup $GF(q)^*$. Cryptographic schemes based on ECC rely on scalar multiplication of elliptic curve points. Scalar multiplication of elliptic curve points can be computed efficiently using the addition rule together with the double-and-add algorithm or one of its variants.

Here we will implement, the scalar multiplication k^P which we obtain by repeating the elliptic curve addition k times, by following the same additive rules and a generator point:

$$(x_2, y_2) = (0, 2) \partial (0, 2)$$

$$= (13, 12) = 2P$$

$$\lambda = \frac{3x_1^2 + a}{2y_1} \bmod p = \frac{3(0)^2 + 1}{2(2)} \bmod 23 = \frac{1}{4} \bmod 23 = 4^{-1} \bmod 23 = 6 \bmod 23$$

$$x_3 = \lambda^2 - x_1 - x_2 \bmod p = (6)^2 - 0 - 0 \bmod 23 = 13$$

$$y_3 = \lambda (x_1 - x_3) - y_1 \bmod p = 6(0 - 13) - 2 \bmod 23 = -11 \bmod 23 = 12$$

$$(x_3, y_3) = (13, 12) \partial (0, 2)$$

$$= (11, 9) = 3P$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod 23 = \frac{2 - 12}{0 - 13} \bmod 23 = \frac{-10}{-13} \bmod 23 = \frac{10}{13} \bmod 23 = 32$$

$$x_3 = (\lambda^2 - x_1 - x_2) \bmod 23 = (22^2 - 13 - 0) \bmod 23 = 471 \bmod 23 = 11$$

$$y_3 = \lambda (x_1 - x_3) - y_1 \bmod 23 = 22(13 - 11) - 12 \bmod 23 = 9$$

$$(x_4, y_4) = (11, 9) \partial (0, 2)$$

$$= (1, 12) = 4P$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod 23 = \frac{2 - 9}{0 - 11} \bmod 23 = \frac{-7}{-11} \bmod 23 = \frac{7}{11} \bmod 23 = 9$$

$$x_4 = (\lambda^2 - x_1 - x_2) \bmod 23 = (9^2 - 11 - 0) \bmod 23 = 1$$

$$y_4 = \lambda (x_1 - x_3) - y_1 \bmod 23 = 9(11 - 1) - 9 \bmod 23 = 12$$

$$(x_5, y_5) = (1, 12) \partial (0, 2)$$

$$= (7, 20) = 5P$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod 23 = \frac{2 - 12}{0 - 13} \bmod 23 = \frac{-10}{-1} \bmod 23 = 10$$

$$x_5 = (\lambda^2 - x_1 - x_2) \bmod 23 = (10^2 - 0 - 1) \bmod 23 = 99 \bmod 23 = 7$$

$$y_5 = \lambda (x_1 - x_3) - y_1 \bmod 23 = 10(1 - 7) - 12 \bmod 23 = -3 \bmod 23 = 20$$

similarly

$(x_6, y_6) = (7, 20) \partial (0, 2) = (9, 11) = 6P$
$(x_7, y_7) = (9, 11) \partial (0, 2) = (15, 6) = 7P$
$(x_8, y_8) = (15, 6) \partial (0, 2) = (14, 5) = 8P$
$(x_9, y_9) = (14, 5) \partial (0, 2) = (4, 7) = 9P$
$(x_{10}, y_{10}) = (4, 7) \partial (0, 2) = (22, 5) = 10P$
$(x_{11}, y_{11}) = (22, 5) \partial (0, 2) = (10, 5) = 11P$
$(x_{12}, y_{12}) = (10, 5) \partial (0, 2) = (17, 9) = 12P$
$(x_{13}, y_{13}) = (17, 9) \partial (0, 2) = (8, 15) = 13P$
$(x_{14}, y_{14}) = (8, 15) \partial (0, 2) = (18, 9) = 14P$
$(x_{15}, y_{15}) = (18, 9) \partial (0, 2) = (18, 14) = 15P$
$(x_{16}, y_{16}) = (18, 4) \partial (0, 2) = (8, 8) = 16P$
$(x_{17}, y_{17}) = (8, 8) \partial (0, 2) = (17, 14) = 17P$

$(x_{18}, y_{18}) = (17, 14)\partial(0, 2) = (10, 18) = 18P$
$(x_{19}, y_{19}) = (10, 18)\partial(0, 2) = (22, 18) = 19P$
$(x_{20}, y_{20}) = (22, 18)\partial(0, 2) = (4, 16) = 20P$
$(x_{21}, y_{21}) = (4, 16)\partial(0, 2) = (14, 18) = 21P$
$(x_{22}, y_{22}) = (14, 18)\partial(0, 2) = (15, 17) = 22P$
$(x_{23}, y_{23}) = (15, 17)\partial(0, 2) = (9, 12) = 23P$
$(x_{24}, y_{24}) = (9, 12)\partial(0, 2) = (7, 3) = 24P$
$(x_{25}, y_{25}) = (7, 3)\partial(0, 2) = (1, 11) = 25P$
$(x_{26}, y_{26}) = (1, 11)\partial(0, 2) = (11, 14) = 26P$
$(x_{27}, y_{27}) = (11, 14)\partial(0, 2) = (13, 11) = 27P$
$(x_{28}, y_{28}) = (13, 11)\partial(0, 2) = (0, 21) = 28P$

and finally

$(x_{29}, y_{29}) = (0, 21)\partial(0, 2) = O$

since

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \bmod 23 = \frac{2 - 12}{0 - 0} \bmod 23 = O$$

The cost of elliptic arithmetic operations: The dominant cost operation in elliptic curve cryptographic schemes is scalar point multiplication, namely computing kQ where, Q is an elliptic curve point and k is an integer. This operation is the additive analogue of the exponentiation operation a^k in a general (multiplicative-written) finite group. The basic technique for exponentiation is the repeated square-and-multiply algorithms.

For the speed of the elliptic curve cryptosystems, the number of elementary field operations for point addition is important. Here we are just interested in "quadratic" field operations, i.e., we do not care about operations which can be done in linear time. One important observation is then the fact that negating a point is "for free", since for any non-zero point $P = (x, y) \in E(F_p)$ the negative point is given as $-P = (x, -y)$. If we count the quadratic field operations of the other basic point operations, we get the following results: Doubling a point takes one multiplication, two squaring and one inversion; adding two different points can be done with one multiplication, one squaring and one inversion in F_q. In practice, the inversion is by far the most time consuming part of these operations (in the computer algebra library LiDIA, one inversion of a random element in a 155-bit prime field takes about the same time as 25 multiplications in this field).

In the development and implementation of elliptic curve cryptography we are interested in the method for computing an equation of the form $m \cdot P$ where, $m \in ¥_{>1}$ and let $P \in E(F_q)$ be a non zero point on some given elliptic curve E. For a positive integer m we let [m] denote the multiplication-by-m map from the curve to itself. This map takes a point P to P+P+...+P (m summands). The notation [m] is extended to $m \leq 0$ by defining [0]P = O and [-m]P = -([m]P). So for instance, as above, [2]P = P+P, [3]P = P+P+P and [-3]P = -(P+P+P). This map is the basis of elliptic curve cryptography. Its properties; computation and uses will be, therefore, the core of this paper.

Basic facts

The frobenius endomorphism: A map which plays a crucial role in the theory of elliptic curves over finite fields is the Frobenius endomorphism. For a curve E/F_q we consider the map:

$$\varphi_q : E(\bar{F}_q) \to E(\bar{F}_q)$$
$$\varphi_q : (x, y) \to (x^q, y^q)$$
$$\varphi_q : O \to O \qquad (18)$$

For two points $P, Q \in E(\bar{F}_q)$ we have $\varphi_q(P+Q) = \varphi_q(P)+\varphi_q(Q)$. It is easy to see that the set $E(F_p)$ is the set of points which are invariant under φ_q, i.e., $P \in E(\bar{F}_q)$ we have $P \in (F_q) \Leftrightarrow \varphi_q(P) = P$.

The Frobenius endomorphism $\varphi_q : E(\bar{F}_q) \to E(\bar{F}_q)$ satisfies the characteristic equation:

$$\varphi_q^2 - t\varphi_q + q = 0 \qquad (19)$$

which can be interpreted in the sense of operators which acts on the group of points of E:

$$(x^{q^2}, y^{q^2}) - t\cdot(x^q, y^q) + q\cdot(x, y) = 0,$$
$$\forall (x, y) \in E(\bar{F}_q) \qquad (20)$$

The integer t which appears in this formula is called trace of Frobenius and is denoted by $Tr(\varphi_q)$. The following theorem explains the relation between t and the number of points, i.e.,:

Theorem: $\#(E(F_p)) = q+1-Tr(\varphi_q)$ (21)

The group of torsion points: Let E/F_q be an elliptic curve. For $n \in ¥_{\geq 2}$ we write $E[n] := E(\bar{F}_q)[n]$ for the group of the n-torsion points of $E(\bar{F}_q)$, i.e., the point $P \in E(\bar{F}_q)$ with nP = O. The structure of the group E[n] is well-known:

$$E[n] \cong Z/n \oplus Z/n \text{ if } \gcd(n, q) = 1$$

Table 3: Point addition/scalar point multiplicative values of P (note that: $kP \equiv P_k$, so that $29P \equiv P_0 \equiv O \equiv (0, 21)$ etc.).

29P = (0)	1P = (0, 2)	2P = (13, 12)	3P = (11, 9)	4P = (1, 12)	5P = (7, 20)
6P = (9, 11)	7P = (15, 6)	8P = (14, 5)	9P = (4, 7)	10P = (22, 5)	11P = (10, 5)
12P = (17, 9)	13P = (8, 15)	14P = (18, 9)	15P = (18, 14)	16P = (8, 8)	17P = (17, 14)
18P = (10, 18)	19P = (22, 18)	20P = (4, 16)	21P = (14, 18)	22P = (15, 17)	23P = (9, 12)
24P = (7, 3)	25P = (1, 11)	26P = (11, 14)	27P = (13, 11)	28P = (0, 21)	

The elliptic group structure: A user of ECC will be interested to know the structure of the group, especially he would like to know the number of the points, $\#E(F_p)$. For example: If he knows $\#E(F_p)$ he knows whether the Pohlig-Hellman[37] attack can be applied. In the Diffie and Helman[3] key exchange, security is dependent upon the order of the group not being divisible by any small primes. While in the Massey-Omura system, the order of the group must be known. Order, generally, is an important component to the security or effectiveness of any real cryptographic application of elliptic curves. It is impossible to write down all the points for large q. But it is not necessary at all. The formula of Hesse-Weil[35] can be applied, which reduces the amount of work enormously.

Group order: Let E be the elliptic curve over a finite field F_q. Hasse's theorem states that the number of points on an elliptic curve (including the point at infinity) is $\#E(F_p) = q+1-t$ where $|t| \le 2\sqrt{q}$, $\#E(F_p)$ is called the order of an elliptic curve E and t is called the trace of E[35]. In other words, the order of an elliptic curve $E(F_p)$ is roughly equal to the size q of the underlying field[15].

Group structure: $E(F_p)$ is an abelian group of rank 1 or 2. That is, $E(F_p)$ is isomorphic to $Z_{n1} \times Z_{n2}$, where, n_2 divides n_1, for unique positive integers n_1 and n_2. Here Z_n denotes cyclic group of order n. Moreover, n_2 divides q-1. If $n_2 = 1$, then $E(F_p)$ is said to be cyclic. In this case $E(F_p)$ is isomorphic to Z_{n1} and there exists a point $P \in E(F_p) = \{kP: 0 \le k \le n_1-1\}$; such a point is called a generator of $E(F_p)$.

Cyclic elliptic curve[38]: Consider again the example above of $E_{1,4}(F_{23})$:

$$y^2 = x^3+x+4 \bmod 23$$

For example, $P_5 = (7, 20) \in E_{1,4}(F_{23})$ because:

$$7^3+7+4 = 354 \equiv 9 \equiv 3^2 \pmod{23}$$

The above points P_i listed in Table 3 can be numbered in such way that:

$$P_i+P_j+P_{i+j} \text{ (indices mod 29)}$$
$$\text{(i.e., } kP \equiv P_k \text{ and } 29P \equiv P_0\text{)} \quad (22)$$

Thus, $\#E(F_{23}) = 29$, which satisfies the Hasse bound since:

$$|(23+1)-29| = 5 \le 2\sqrt{23} \cong 9.59$$

Since $\#E(F_{23}) = 29$, which is prime and which is found by counting all the points, also known as naïve method. $\#E(F_{23})$ is cyclic and any point other than O is a generator of $\#E(F_{23})$. For example, P = (0, 2) is a generator as shown in Table 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 $nP = O$ and $mP \neq O$ for any integer m 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 $N = \#E(F_p)$ 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 $\ell = \#E(F_p)/n$ will not be divisible by n. Choose a random point $Q \neq O$ on the elliptic curve E, then verify whether the point $P = \ell Q$ has order n. This can be done simply by checking that $nP = O$. (Since n is prime, there is no other positive integer $m < n$ such that $mP = O$) If it is true, then $P = \ell Q$ is the point we need; otherwise, choose another point Q and repeat. This is the technique deployed by D. Shank in the Shank's algorithm which we will discuss later in the text.

The mechanics of order of points: Let $E(F_p)$ denote the set of rational points on E. It is easy to see that the number of points in $E(F_p)$ with X-coordinate $x \in F_p$ is 0, 1, or 2. More precisely, there are:

$$1 + \left(\frac{x^3 + ax + b}{p}\right) \quad (23)$$

rational points on E with X-coordinate equal to x. Here $\left(\frac{\cdot}{p}\right)$ denotes the quadratic residue symbol (also known as Legendre symbol). Including the point at infinity, the set of rational points $E(F_p)$ of E, therefore, has cardinality:

$$\sharp E = 1 + \sum_{x \in F_p} \left(1 + \left(\frac{x^3 + ax + b}{p} \right) \right) = 1 + p + \sum_{x \in F_p} \left(\frac{x^3 + ax + b}{p} \right) \quad (24)$$

When x^3+ax+b is a square modulo p then two points are contributed to the order. When $p|x^3+ax+b$ one point is added. If x^3+ax+b is not a square or divisible by p then nothing is contributed in the sum. This implies that evaluating the sum:

$$\sum_{x \in F_p} \left(\frac{x^3 + ax + b}{p} \right) \quad (25)$$

is the same problem as computing $\#E(F_p)$. For very large primes ($p < 200$ say) a straightforward evaluation of this sum is an efficient way to compute $\#E(F_p)$. In practice it is convenient to make first a table of squares modulo p (as was done in Table 3) and then count how often x^3+ax+b is a square for x = 0, 1,....., p-1. The running time of this algorithm is $O(p^{1+\varepsilon})$ for every $\varepsilon>0$.

Many particular elliptic curves over particular finite fields, whose 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 $a \neq 0 \pmod p$ or $E_p(0, b)$: $y^2 = x^3+b$, for $b \neq 0 \pmod p$[42].

For a general elliptic curve over larger finite field F_p, one should use Shanks' Baby-step-Giant-step strategy[39]. Alternative techniques are also available to compute the exact order quickly, these are: Schoof's algorithm used to compute the order of an elliptic curve over very large finite field; reduction algorithm when the endomorphism ring of E is known.

Shanks' method (baby-step, giant-step): The idea behind Shanks' algorithm[39] is to pick up a random point P on the elliptic curve and to compute an integer m such that: $p+1-2p^{1/2} \le m \le p+1\ 2p^{1/2}$ and $mP = 0$. 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.

The elliptic curve version of his algorithm is as follows: Assume that Q is a generator of order n and given point A, we want to compute k such that kQ = A. Let $m = \lfloor \sqrt{n} \rfloor$. Store in a list L_1 the pairs (j, jmQ) for $0 \le j \le m-1$. Sort this list by the second element of the pairs. Create a second list L_2 from the pairs (I, -iq+A) for $0 \le i \le m-1$. Sort this list by the second element. Search the list until pairs (j, p) $\in L_1$ and (j, p) $\in L_1$ are found. We have:

$$P = jmQ$$
$$P = -iQ+A$$
$$jmQ = -iQ+A$$
$$(jm+i)Q = A$$

Thus, the multiple of Q such that kQ = A is k = jm+i. Of course, A must be a multiple of Q for this algorithm to work. One simple way to be certain of this is to choose an elliptic curve with order prime p. Since the order of any element must divide the order of the group, every element must have either order 1, namely the identity, or order p. The space requirement is for storage of the two lists, each composed of $n^{1/2}$ points. An algorithm of Daniel Shanks reduces running time $O(n^{1/2} \log n)$ and the space $O(n^{1/2})$ points.

The time complexity is dominated by the sorting of these $n^{1/2}$ elements. Its running time is $O(p^{1/4+c})$ for every $\varepsilon>0$. This algorithm becomes impractical for somewhat larger primes; it becomes impractical when p has more than, say, 20 decimal digits, or when multiple values of m are available for every point p. However, Mestre[40] have developed a new algorithm which showed that if Shanks' algorithm fails for an elliptic curve, then it will not fail on its twisted curve.

Implementation of Shanks' Method (baby-step, giant-step): As an example consider our earlier elliptic curve: $y^2 = x^3+4 \bmod 23$. This curve has order 29. Choose as a generator Q = (0, 2) and the element of unknown index A = (18, 9). Let $m = \lfloor \sqrt{29} \rfloor = 6$

List 1 = {(1, (9, 11), (2, (17, 9), (3, (10, 18), (4, (7, 3), (5, (0, 2)}

List 2 = {(1, (8, 15), (2, (17, 9), (3, (10, 5), (4, (22, 10), (5, (16, 19)}

Item 2 in L_1 matches item 2 in L_2. This indicated that: 12Q = -2Q+A or 14Q = A which from Table 3, we have A = 14Q = 14P = (18, 9) which is the same point we started with and so the order of point A is k = jm+i = 14. This is correct.

If an elliptic curve has Complex Multiplication properties, there are other efficient algorithms to count the points[41-43]. For larger values of p one needs the help of an algorithm developed by Schoof[44]. 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.

Pohlig and Hellman's method: This method (also referred to as Silver -Pohlig-Hellman's method)[37] reduces the problem to a determination of m mod 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 $n = p_1^{e_1} K P_s^{e_s}$ be the order of g. Let p be any prime in the set $\{p_1, K, p_s\}$ Then

$z = m(\mathrm{mod}\ p^{e}) = a_0 + a_1 p + K + a_{e-1} p^{e-1}$, 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, $O(\sum e_i (\log n + p_i))$ 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 and Hellman's algorithm, n should be divisible by a large prime number (>280), or indeed, n must be prime >280 for the maximum security possible.

The Schoof-Elkies-Atkin (SEA) algorithm: It is obvious that if we can calculate the trace $Tr(F_q)$ of Frobenius, then we have information about E/F_q. The calculation of Frobenius is the kernel of the SEA-algorithm. Recent methods are based on the work of the Dutch mathematician Schoof[41] and were improved by Atkin and Elkies in late 90s at the end of the last century. Advanced methods from arithmetical algebraic geometry (modular forms) are used intensively. Finally this method became to be known as Schoof-Elkies-Atkin (SEA) algorithm[45]. SEA-algorithm can be used in order to calculate the number of points on an elliptic curve over F_p for large value of p.

Schoof studied for a prime $\ell \neq p$ the torsion group $E[\ell]$. Then he considered how Frobenius endomorphism works on ℓ. The characteristic equation of Frobenius can be reduced modulo ℓ. We have:

$$\varphi_q^2 - t_\ell \varphi_q + q^\ell = 0 \qquad (26)$$

where, $t_\ell \equiv t \bmod \ell$ and $q_\ell \equiv q \bmod \ell$. This equation can be rewritten as:

$$t_\ell \varphi_q = \varphi_q^2 + q_\ell \qquad (27)$$

We consider a (generic) point y. We calculate the righthandside and lefthandside of the previous equation for this point. We find for the lefthandside:

$$(\varphi_q^2 + q_\ell)(x,y) = \left(x^{q^2} + \frac{f_1(x,y)}{g_1(x,y)}, y^{q^2} + \frac{f_2(x,y)}{g_2(x,y)}\right) \qquad (28)$$

We eliminate the y in the x-coordinate, since $y^2 = x^3+ax+b$, the x-coordinate can be written as the quotient $\frac{F(x)}{G(x)}$. The lefthandside can be rewritten as the quotient $\frac{K_{t_\ell}(x)}{L_{t_\ell}(x)}$. In the lefthandside t_ℓ is unknown. If the correct value of t_ℓ is chosen F(x) and K_{t_1} have a common non-trivial divisor. We have to chose $t_1 = 0, 1, 2, \ldots$ until t_1 fits. Now by varying ℓ we find related values of t_1. The value of t can be calculated using the Chinese Remainder Theorem[2].

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[46] later also explained in much cryptography literature. In short, we partition the group into 3 subsets and then perform the following recursive search: $x_{i+1} = x_{iu}$, 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[47,48]. 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, i.e., 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[49] 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 is also a method proposed by Silverman and Stapleton[50], to solve multiple discrete logarithms: $\log_g y_1,\ldots, \log_g y_M$. This method was originally used to attack the ECDLP.

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 and Paulus[51] discussed that the sieving methods are still not efficient on ECDLP yet "lifting" points on $E(GF(p^n))$ to points on an elliptic curve, $E\%(¤)$ where, (¤) is the rational field. That is, given $P \in E(GF(p^n))$, find an elliptic curve. $\tilde{E}(¤)$ and a point $Q \in \tilde{E}(¤)$ such that $Q \equiv P \pmod p$. The natural candidate for a factor base is a set of points of small height on $E\%(¤)$.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[52] and Suzuki[52,53] 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.

Public-key Cryptography (PKC): Diffie and Hellman[3] were the first to introduce the public-key cryptography to overcome the difficulty of key distribution encountered with the private-key cryptosystems. This protocol later become to be known as Diffie-Hellman (DH) key exchange. In the later years other public-key cryptosystems were developed for various different cryptographic application. Among these we can mention the RSA[5], Digital Signature Algorithm (DSA) [6,7]and ElGamal[8] cryptosystem. The invention of Elliptic Curve Cryptography (ECC) in 1985 offered a new level of security for public key cryptosystems[14-16], which provide both encryption and digital signatures services using already existing public-key algorithms.

It was Miller who first proposed the Diffie-Hellman key exchange protocol[8] on the bases of elliptic curve[13]. The elliptic curve based analogues of the ElGamal scheme and the Massey-Omura scheme[12,30] was proposed by Koblitz[12,30]. The first elliptic curve based analogue to the RSA scheme was introduced in 1991[54]. A more advanced elliptic curve based analogue to the RSA scheme was introduced by Demytko[55].

At the foundation of every public-key cryptosystem is a hard mathematical problem that is computationally infeasible to solve. For example, RSA and Diffie-Hellman rely on the hardness of integer factorization and Discrete Logarithm Problem (DLP), respectively[24,25,29]. Unlike these cryptosystems which operate over integer fields, the Elliptic Curve Cryptosystems (ECC) operates over points on an elliptic curve.

The mechanics of Discrete Log Problem (DLP): One of the important tool necessary for the implementation of public-key cryptosystems is the Discrete Log Problem (DLP). In an (abelian) group G (multiplicatively written) we can consider the equation $y = x^n$, $x, y \in G$, $n \in Z$. If x and y are known real numbers and it is also known that x is some power (say, n) of y, then logarithms can be used to find (n("$= \log_x (y)$") in an efficient manner. However, if x and y are given such that, $y = x^n = \underbrace{x \cdot x \ldots \cdot x}_{n \text{ times}}$, then in general it is technically much harder and hence the determination of n cannot be carried out in a reasonable amount of time. This is equivalent to the well-known real logarithm, we call n the discrete logarithm of y related to the base x[56,57].

Remark 1: The operation "exponentiation" $x \to y := x$" can be implemented in as a quick, efficient algorithm.

Remark 2: If p is a prime number, then Z_p denotes the set of integers {0, 1, 2,..., p-1}, where addition and multiplication are performed modulo p. It is well-known that there exists a non-zero element $g \in Z_p$ such that each non-zero elements in Z_p can be written as a power of g; such that an element Z_p is called a generator of Z_p.

Remark 3: The Discrete Logarithm Problem (DLP) is the following: given a prime p, a generator g of Z_p and a non-zero element $\beta \in Z_p$, find the unique integer k, $0 \le k \le p-2$, such that $\beta = g^k \bmod p$. The integer k is called the discrete logarithm of β to the base g.

Many popular modern cryptosystems base their security on the Discrete Logarithm Problem (DLP). Based on the difficulty of this problem, Diffie and Hellman[3] proposed the well-known Diffie-Hellman key agreement scheme in 1976. Since then, numerous other cryptographic protocols whose security depends on the DLP have been proposed, including: the ElGamal encryption and signature scheme[8], the U.S. government digital signature (DSA)[6,7] is perhaps the best known example of a DLP system, the Schnorr signature scheme[57] and the Nyberg-Reuppel signature scheme[58]. Due to interest in these applications, the DLP has been extensively studied by mathematicians for the past 20 years. The mathematical challenge here lies in computing discrete logarithms in finite fields of type Z_p, which consist of the integers modulo a large prime p. Although this problem can be considered difficult, there are known sub-exponential time algorithms for solving it, such as the number field sieve. In practical terms, sub-exponential time means that a determined hacker with enough processing power can break the system in a few months.

The corresponding problem in additive (i.e., abelian) groups is: given P and kP (P added to itself k times), find the integer k. This is much more difficult. There is no one-step operation like "taking logarithms" that we can use to get the solution. So we may know P and kP and yet not be able to find k in a reasonable amount of time. This is called the Discrete Log Problem for abelian groups. We could always repeatedly subtract P from kP till we got O. But if k is large, this will take us a very long time. Several important cryptosystems are based on the difficulty of solving the DLP over finite abelian groups. The solution is even tougher if the underlying group arises from an elliptic curve over a finite field F_q.

Elliptic Curve Discrete Logarithm Problem (ECDLP): The fundamental mathematical operation in RSA and Diffie-Hellman is modular integer exponentiation[59]. However, the core of elliptic curve arithmetic is an operation called scalar point multiplication, which computes Q = kP (a point P multiplied k times resulting in another point Q on the curve)[57]. For example, 11P can be expressed as 11P = (2*((2*(2*P))+P. Furthermore, when a point P on an elliptic curve E is given, there is a minimum positive integer n such that nP = O, the identity point or point at infinity. Integer n is called the order of the point P. It is known that n is a divisor of the order of the curve E. Consider our earlier comment that these scalar multiples form a subgroup of points <P>. Here <P> is the finite cyclic group <P> = {P, 2P, 3P,..., nP}, with order n. The problem of calculating k from a given points P and Q is called "the discrete logarithm problem over the elliptic curve (ECDLP)"

We now give a more precise definition of the ECDLP: Given an elliptic curve $E(F_q)$, point $P \in E(F_q)$ of order n and a point $Q \in E(F_q)$, determine the integer k $0 \le k \le n-1$, such that Q = kP, provided that such an integer exists. Here Q is the public-key and k is the private-key. Thus, the ECDLP is based on the intractability of scalar multiplication of points on elliptic curves. Note that the number k is called the "discrete log of Q base P", which is why this is referred to as the Elliptic Curve Discrete Logarithm Problem. Point P is called the base point in this problem. We call $k = \log_p Q$ the elliptic curve discrete logarithm of Q to the base point P.

Note that we can easily calculate Q = kP from given k and P, but it is computationally difficult to calculate the scalar k from points Q and P. The security of ECC relies on the hardness of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP), which states that given P and Q = kP, it is hard to find k[11]. 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 that has some points (base point G) which has large prime order n, and $\#E(F_q) = n \cdot p$, where h is a small integer. To date, the most efficient general algorithm to solve the ECDLP is Pollard-ρ, which has $\sqrt{\pi n}/(2r)$, where r is the parallelized processor number. The runtime is exponential in n.

While a brute-force approach is to compute all multiples of P until Q is found, k would be so large in a real cryptographic application that it would be infeasible to determine k in this way. If a prime p as large as 160-bits long is selected, we cannot find k within a reasonable time, even if we use the most efficient algorithms known so far with the world's most powerful computers.

Based on the difficulty of this problem, Koblitz[12] and Miller[13] independently in 1985 proposed using the group of points on an elliptic curve defined over a finite field to implement the various discrete log applicable to existing public-key cryptosystems. One

such cryptographic protocol that is being standardized by a accredited standards organizations is the elliptic curve analog of the digital signature algorithm (DSA), called elliptic curve digital signature algorithm (ECDSA), which we will discuss later in the text.

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.

Key exchange protocol in PKC: The mathematical idea fundamental to public-key cryptography is that of a hard problem and from such problems, mechanisms for public-key key exchange might be constructed[6], if an additional technical requirement (a trapdoor) can be designed then the hard problem might possibly be used to constructed a public-key encryption or a digital signature algorithm[6]. Today the candidate hard problems which apply for public-key cryptography is discrete logarithm problem over a finite field[29] and integer factorization[25]. It is often stated that IFP is easier to state and understand than the ECDLP. However, this may be true, it is important to note that IFP has not been seriously studied by a multitude of people over thousands of years, there have been only two major advances in techniques for solving IFP: the quadratic sieve factoring algorithm (together with its predecessor, the continued fraction factoring algorithm) and the number field sieve. The latter algorithm involves some sophisticated mathematics (especially algebraic number theory) and is only completely understood by a small community of theorists. Furthermore, prior to computer age, mathematicians were constrained to looking for algorithms for Integer Factorization Problem (IFP) that were efficient by "hand" rather than by large networks of computers. While in case of ECDLP, a fact that is commonly overlooked is that much of the work done on the DLP prior to 1985 also applies to the ECDLP - i.e., because the ECDLP can be viewed as being the same as the DLP but in a different algebraic setting.

In RSA, public and private keys denoted by {e, n} and {d, n}, respectively, are generated by relying on the IFP[60] which, generally stated, is: Given a large number n which is a product of two prime numbers p and q: n =pq, it is difficult to determine the two prime factors by only knowing n. These primes are then used to generate the keys[2] . The most common way of trying to break this system, is by trying to factor n (from the public-key) to obtain the two primes from which the private-key can be determined[61]. This is very costly. Unfortunately, encrypting a message M, involves exponentiation, $C = M^e$, modn, which involves a lot of computation.

Keys in ECC are generated by relying on the Elliptic Curve Discrete Logarithm Problem (ECDLP) which is the problem of determining an integer k (provided it exists) such that kP = Q where P and Q are points on the elliptic curve[62]. ECC does not involve exponentiation, but uses multiplication, hence it is not as computationally costly as RSA. Also, the underlying mathematical problem of ECC, ECDLP is fully exponential, whereas sub-exponential algorithms exist for IFP used in RSA[63]. Because it is more 'infeasible' to solve ECDLP the same level of security, can be provided with shorter keys in ECC compared with RSA.

Menezes and Johnson[63] gives comparisons of the difficulty of these problems. His conclusion is that the ECDLP is computationally more difficult than the DLP or the IFP, hence, cryptosystem based on elliptic curve can be as secure as their more traditional counterparts with significantly smaller fields. The expected time to solve the ECDLP with a point P as the generator having order of a 160-bit prime is approximately equal to the time required to solve the DLP with a 1024 bit modulus or to factor a 1024 bit n into primes p and q. These estimates are based on the currently best known algorithms for the problems: for the ECDLP this is Pollard's-ρ method, for the DLP and the IFP the best algorithm is the number field sieve. The reason for the large difference in field size for the different systems is because of the running times of the algorithms for the problems. No sub-exponential time algorithm for the ECDLP has been found that applies to general curves; the fastest for the DLP are not applicable in the elliptic curve setting. Due to these factors, ECC is better suited for low bandwidth, computational power and memory situations especially in mobile and wireless environment[55].

Elliptic curve encryption algorithm: Elliptic curve cryptography can be used to encrypt plaintext message, M, into ciphertexts. The plaintext message M is encoded into a point P_M from the finite set of points in the elliptic group, $E_p(a, b)$. The first step consists in choosing a generator point, $G \in E_p(a, b)$ such that the smallest value of n for which nG = O is a very large prime number. The elliptic group G and the generator G are made public.

Assume that Bob and Alice intends to communicate. Each user selects a private key and uses it to compute their public-key. For example, Alice (A) selects a private-key $n_A<n$ and computes the public-key P_A as: $P_A = n_A G$. To encrypt the message P_M for Bob (B), Alice chooses a random integer k and computes the ciphertext pair of points P_C using Bob's public-key P_B:

$$P_C = [(kG), (P_M+k P_B)]$$

After receiving the ciphertext pair of points, P_C, Bob multiplies the first point, (kG) with his private-key, n_B and then subtracts the result to the second point in the ciphertext pair of points, $(P_M+k P_B)$:

$$(P_M+k P_B)-[n_B(kG)] = (P_M+k n_B G)-[n_B(kG)] = P_M$$

which is the plaintext point, corresponding to the plaintext message M. Only Bob, knowing the private-key n_B, can remove $n_B(kG)$ from the second point of the ciphertext pair of point, i.e., $(P_M+k P_B)$ and hence retrieve the plaintext information P_M.

Implementation of elliptic curve encryption scheme: The elliptic curve group generated by our earlier elliptic curve i.e., $E_p(a, b) = E_{23}(1, 4)$. Since $\#E(F_{23})$, which is prime. $E(F_{23})$ is cyclic and any point other than O is a generator of $E(F_{23})$. For example, $G \equiv P = (0, 2)$ is a generator point such that the multiples kG of the generator point G are (for $1 \le k \le 29$), as shown in Table 3.

If Alice wants to send to Bob the message M which is encoded as the plaintext point $P_M = (7, 3) \in E_{23} (1, 4)$. She must use Bob's public-key to encrypt it. Suppose that Bob's secret-key is $n_B = 6$, then his public-key will be:

$$P_B = n_B G = 6(0, 2) = 6P = (9, 11)$$

Alice selects a random number $k_A = 15$ and Bob's public-key $P_B = (9, 11)$ to encrypt the message point into the ciphertext pair of points:

$$\begin{aligned} P_C &= [(k_A G), (P_M+k_A P_B)] && \text{(since: } k_A P_B = 15P_B = 15(6P) = 90P = 3P\text{)} \\ &= [15(0, 2), (7, 3)+15(9, 11)] \\ &= [15P, 24P+3P] && \text{(since: } 24P+3P = 27P = (13, 11)\text{)} \\ &= [(18, 14), (13, 11)] \end{aligned}$$

Upon receiving the ciphertext pair of points, $P_C = [(18, 14), (13, 11)]$, Bob uses his private-key, $n_B = 6$, to compute the plaintext point, P_M, as follows:

$$\begin{aligned} (P_M+k_A P_B)-[n_B(kG)] &= (13, 11)-[6(18, 14)] \\ &= (13, 11)-[6(15P)] \\ &= (13, 11)-3P] && \text{(since: } -P = (x_1, -y_1)\text{)} \\ &= 27P+26P = 24P = (7, 3) && \text{(since: } -3 \equiv 26 \bmod 29\text{)} \end{aligned}$$

or

$$(P_M+kP_B)-[n_B(kG)] = (7, 3)$$

and which maps the plaintext point $P_M(7, 3)$ back into the original plaintext message M.

The mechanics of Elliptic Curve Diffie-hellman (ECDH) key exchange scheme: The elliptic curve Diffie-Hellman scheme is a key agreement scheme based on ECC. It is designed to provide a variety of security goals depending on its application - goals it can provide include unilateral implicit key authentication, mutual implicit key authentication, known-key security and forward secrecy-depending on issues like whether or not public keys are exchanged in an authentic manner and whether key pairs are ephemeral or static[64,65].

Besides the curve equation, an important elliptic curve parameter is the base point, P, which is fixed for each curve[59]. In the Elliptic Curve Cryptosystem, the large random integer k is kept private and forms the secret-key, while the result Q of multiplying the private-key k with curve's base point P serves as the corresponding public-key.

In ECDH key agreement, two communicating parties, say Alice (A) and Bob (B) agrees to use the same curve parameters[31,66]. They generate their private keys, k_A and k_B and corresponding public-keys $Q_A = k_A P$ and $Q_B = k_B P$. The parties exchange their public-keys. Finally, each multiplies their private-key and the others public-key to arrive at a common shared secret-key $k_A Q_B = k_B Q_A = k_A k_B P = S_{AB} P$, where, $k = k_A k_B$ The session-keys are the same (since $S_{AB} = S_{BA} = k$); hence Alice and Bob now have a shared secret-key k. Future communications occur using the session key k. Thus, k

can be used as a session-key for a private-key encryption algorithm such as Data Encryption Standard (DES) or AES, or any other cryptographic protocol[2].

The protocol depends on the discrete logarithm problem for its security. It assumes that it is computationally infeasible to calculate the shared secret key, $S_{AB}P$, given the two public values $Q_A = k_AP$ and $Q_B = k_BP$ when the prime p is sufficiently large.

If Eve (the eavesdropper) can recover S_{AB} from this data then Eve is said to have solved Diffie-Hellman Problem (DHP). However, it turns out that if the numbers are all sufficiently large, it is very hard to calculate the discrete logarithm in a reasonable time. The security of the Diffie-Hellman algorithm depends on this fact. It is believed for most groups in use in cryptography that DHP and the DLP are equivalent[67,68], in complexity-theoretic sense (there is a polynomial time reduction of one problem to the other and vice versa. Meyer[69] has shown that breaking the Diffie-Hellman protocol was equivalent to computing discrete logarithms under certain assumptions.

The base point P for this system is one with large order in E. If the subgroup generated by P is small then the secret-key is being chosen from a small keyspace and the systems is vulnerable to keyspace search. The points on the curve E do not need to be generated by P; it may be the group is not even cyclic. It can be shown, however, that the group is the product of two cyclic groups[15]. Additionally, the order of E should have no small prime factors. If it does, then methods exist that can exploit this to solve this discrete logarithm problem.

Implementation of ECDH algorithm: As a simple example, let's use our earlier elliptic curve i.e., is $E_p(a, b) = E_{23}(1, 4)$. Alice (A) chooses the secret-key $k_4 = 4$ and computes her public-key (Table 3 and Eq. 22):

$$Q_A = k_AP = 4P_1 = P_1+P_1+P_1+P_1 = P_4 = (1, 12)$$

Similarly, Bob (B) chooses the secret-key $k_B = 6$ and computes his public-key (Table 3):

$$Q_B\ k_BP = 6P_1 = P_1+P_1+P_1+P_1+P_1+P_1 = P_6 = (9, 11)$$

Thus, their common secret-key S_{AB}, i.e.,:

Alice computes: $k_AQ_B = 4P_6 = P_6+P_6+P_6+P_6 = P_{24} = (7, 3)$
and Bob computes: $k_BQ_A = 6P_4 = P_{24} = (7, 3)$
Such that: $S_{AB} = k_AP_B = k_BP_A = P_{24} = (7, 3)$

An alternative form of ECDH is the Elliptic curve decision Diffie-Hellman problem (ECDDHP)[70]. The ECDDHP is stated as follows: Given a point p of order n in an elliptic curve E over a finite field F_q and three points (kP), (ℓP) and (mP), the ECDDHP is to decide whether $m = k\ell$ (modulo the order of point P). The ECDDHP is not harder than the ECDHP. Boneh and Venkatesan[71], Boneh and Shparlinski[72] discussed many issues on security of the ECDHP and related schemes.

ElGamal for elliptic curves: Elgamal[8] was the first mathematician to propose a public-key cryptosystem based on the Discrete Logarithm problem. He in fact proposed two distinct cryptosystems: one for encryption and the other for digital signature scheme in 1984, well before elliptic curves were introduced in cryptography. Since then, many variations have been made on the digital signature system to offer improved efficiency over the original system. The ElGamal public-key encryption scheme can be viewed as Diffie-Hellman key agreement protocol in key transfer mode. Its security is based on the intractability of the discrete logarithm problem and the Diffie-Hellman problem.

Instead of Diffie-Hellman, Alice and Bob can also decide to use the protocol of ElGamal. This protocol does not create common key, but using ElGamal a message can be sent from Alice to Bob.

Alice and Bob agree to a prime p, a curve E and a point P (generator). Bob (B) generates a secret-key k_B such that $1 \le k_B \le p-2$. and publishes $B = k_BP$, i.e., (E, P, B) are public-key system.

Encryption: Alice (A) has a message m which has to be sent and calculates the related point M. Alice then generates a random number k_A such that $1 \le k_A \le p-2$. She then computes $A = k_AP$ and ciphertext $C = M+k_AB$ and sends (A, C).

Decryption: Bob receives the message (A, C).
He then computes: $M = C-k_BA = (M+k_Ak_BP)-k_Ak_BP$

It is not trivial to find a point M for the message m. It is impossible to choose for m the x-coordinate of M. In about 50% of the cases m will not appear as x-coordinate. This problem can be avoided by adding a few bits to m, which can be chosen arbitrarily. If an x-coordinate is known, then it is easy to calculate the related y.

Implementation of ElGamal cryptosystem for elliptic curves: As a simple example, Bob (B) chooses public-key set: $B = k_BP = 10P = (22, 5)$, where the secret-key $k_B = 10$ (taken from Table 3).

Let Alice select message point: M = (7, 3) (a point on E) and selects a random secret-key $k_A = 6$, such

that; $A = k_A P = 6P = 6\ (0.2) = (9, 11)$ and encrypts the message:

$$C = M + k_A B = (7, 3) + 6(10P) = 24P + 2P = 26P = (11.14)$$

Decryption: Bob receives the message (A, C) = (6P, 26P). He then computes: $M = C - k_B A = (11, 14) - 10(6P)$ which decrypts to the original message m chosen point M = (7, 3).

The ElGamal signature algorithm is similar to the encryption algorithm in that the public-key and private-key have the same form; however, encryption is not the same as signature verification, nor is decryption the same as signature creation as in RSA. The DSA is based in part on the ElGamal signature algorithm.

For a long period of time (1985-1996) after the birth of the ElGamal signature scheme and the family of such signatures (e.g., Schnorr and DSS), it was widely believed that the difficulty of factoring such a signature should somehow be related to the discrete logarithm in a large subgroup of a finite field.

However, no formal evidence (formal proof) was ever established until 1996. When Poincheval and Stern succeeded in demonstrating affirmative evidence for relating the difficulty of signature forgery under a signature scheme in the ElGamal-family signatures to that of computing discrete logarithm[73]. They do so by making use of a powerful tool: the random oracle model (ROM) for proof of security[22]. The ROM-based technique of Pointcheval and Stern is an insightful instantiation of the general ROM-based security proof technique to proving security for the ElGamal-family signatures.

Massey-Omura elliptic curve-PKC: Massey-Omura is another public-key cryptosystem[74]. Assume that the curve E/F_q and the order n of E are public. Alice and Bob want to communicate. Each selects a random secret integer $e \in Z_n^*$ (e for encrypt) and computes, $d = e^{-1} \bmod n$, with the extended Euclidean algorithm. Alice selects e_A (and computes d_A); Bob selects e_B (and computes d_A). Alice wishes to send the plaintext message associated with the point P to Bob. She computes $e_A P$ and transmits this quantity. Bob receives this point; note that he cannot at all determine P at this time. He sends back to Alice the quantity $e_B e_A P$. Alice computes $d_A e_B e_A P$. Note that $d_A e_A P \equiv 1 \bmod n$, so Alice has really "unmasked" $e_B P$. She transmits this to Bob, who multiplies by his decryption key to get $d_B e_B P = P$. Quantities that have passed through insecure channels in this scheme are $e_A P$, $e_B e_A P$ and $e_B P$. Solving the discrete logarithm problem given $e_A P$ and $e_B e_A P$ for the quantity e_B would allow the attacker (Eve) to compute d_B and thus decrypt the message.

Implementation of Massey-Omura cryptosystems: Recall elliptic curve group generated by the our earlier elliptic curve is $E_p(a, b) = E_{23}(1, 4)$. Since $\#EF_{23} = 29$, with $G \equiv P = (0, 2)$as the generator point such that the multiples kG of the generator point G are (for $1 \le k \le 29$), are as shown in Table 3.

Alice selects: $e_A = 6$ and computes $d_A = 6^{-1} \bmod 29 = 5$
Bob selects: $e_B = 13$ and computes $d_B \equiv 13^{-1} \bmod 29 = 9$
Alice needs to transmit message point: M = (14, 18) = 21P to Bob.
She transmit: $e_A M = 6\ (14, 18) = 6(21P) = 10P = (22, 5)$
Bob replies with: $e_A e_B M = 13(22, 5) = 13(10P) = 14P = (18, 9)$
Alice sends back to Bob: $d_A e_B e_A M = 5(18, 9) = 5(14P) = 12P = (17, 9)$
Bob now multiplies this value by his decryption key: $d_B(d_A e_B e_A M) = 9(17, 9) = 9\ (12P) = 21P = (14, 18) = M$
Bob recovers back the message point: M= (14, 18)

Menezes-Vanstone elliptic curve cryptosystem: 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 n with the secret key Q = dP. The pair (P, Q) are public keys[75,76].

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 point including the point kP and two finite field elements y_1 and y_2 where $y_1 = c_1 m_1 (\bmod p)$ and y_2 and assuming $kQ = (c_1, c_2) \neq (0, 0)$.

Decryption: The receiver uses his secret key d to compute the point d(kP) that should be exactly $kQ = (c_1, c_2)$. Hence, the receiver can recover the message by: $(y_1 c_1^{-1} (\bmod p), y_2 c_2^{-1} (\bmod p)) = (m_1, m_2)$.

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 kP and one finite field element $y = c_1 m (\bmod p)$.

There are other proposals/standards for 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.

RSA algorithm

Key generation: Bob (B) chooses two large prime numbers p, q; n = pq; an integar e_B, $e_B d_B \equiv 1 \bmod \phi(n)$[2,9]. He publishes (n, e_B) as the public-key and keeps (n, d_B) as the secret-key (reveals it no one). Discard securely p and q.

Encryption: $C \equiv M^{e_B} \bmod n$. Here C is the ciphertext of message M to B.

Decryption: $M = C^{d_B} \bmod n$

The product, n, is the modulus, e is the public exponent and, d is the secret exponent. You can publish your public-key freely, because there are no known easy methods of calculating d, p, or q given only (n, e) (your public-key). If p and q are each 1024 bits long, the sun will burn out before the most powerful computers presently in existence to factor your modulus into p and q. Authentication on the other hand, is not as easy to guarantee in public-key cryptography. Since everybody knows everybody else's public-key, Eve can easily send message to Alice claiming to be Bob[2].

Elliptic curve RSA (ECRSA) algorithm: We now mention RSA-type elliptic curve cryptosystems which may be both controversial and interesting to many researchers[14]. There are works on designing such cryptosystems or exploiting the connections with RSA cryptosystems.

Key generation

1. Bob (B) selects two distinct primes p, q with $p \equiv q \equiv 2 \bmod 3$ and computes n = pq;
2. B chooses integers e_B, d_B with $e_B d_B$. (Here we can use (p+1)(q+1) in place of, 1cm (p+1)(q+1)). B publishes e_B as public-key and keeps d_B as secret-key.

Encryption

1. Alice (A) represents the message m as a pair of (m_1, m_2) and regards it as a point M on the elliptic curve E given by $y^2 = x^3+ax+b$, where $b = m_2^2-m_1^2$. (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_B to obtain $C = (c_1, c_2) = e_B M$. Here $C = (c_1, c_2)$ is the ciphertext of message m to B.

Decryption: B computes $d_B C = M \pmod n$ on E to obtain M.

Note: $d_B C = d_B e_B M = (1+k(p+1))\, M = M+k(p+1)\, M = M+\infty = M \pmod p$, with some $k \leq 1$. Similarly, for mod q, we have $d_B C = M \bmod q$. So $d_B C = M \bmod n$. Where, (p+1) M mod p and (q+1) M mod q are ∞, because $E(Z_n)$ and $E(F_p) \oplus E(F_q)$, and $\#E(F_p) = q + 1$. The curve order is in such form because the curve is supersingular mod p and mod q.

Digital Signature Algorithm (DSA): For a digital signature, the main idea is no longer to disguise what a message says, but rather to prove that it originates with a particular sender. The Digital Signature Algorithm (DSA), was proposed in August 1991 by the U.S. National Institute of Standards and Technology (NIST) and was specified in a U.S. Government Federal Information Processing Standards[7] called the Digital Signature Standard (DSS). The DSA[11,77], is based on what's called the discrete logarithm problem, it requires that q be a 160-bit prime and p a prime between 512 and 1024 bits.

The RSA, for example, has a simple way of implementing digital signature algorithm[2]. If the "owner" of a code (n, e) wants to prove that she's the sender of a message M, she can use her private "decoding" exponent d to compute $C = M^d \bmod n$ and then send both M and C. The receiver can then persuade himself that the message truly originated with the owner of d by computing C^e and checking that it's the same as M.

The DSA can be viewed as a variant of the ElGamal signature scheme[8]. Its security is based on the intractability of the Discrete Logarithm Problem (DLP) in prime-order subgroup of Z_p^*. In terms of computational difficulty, the discrete logarithm problem seems to be on a par with factoring. The basic idea of DSA is for the "signer" of message M-that is, the possessor of the value x behind the publicly known, $g^x \bmod p$-to append a pair of numbers r and s obtained by secretly picking another number k between 1 and q, computing $r = (g^k \bmod p) \bmod q$ (i.e., computing $g^k \bmod p$ and then taking the remainder of that number mod q) and $s = k^{-1}(SHA(M)+xr) \bmod q$, where k^{-1} is the multiplicative inverse of k (mod q) and SHA is the Secure Hash Algorithm. Another NIST standard, SHA (official acronym is SHA-1) reduces a character string of any length to a 160 bit string of gibberish. In the DSA, q is a 160 bit prime divisor of p-1 and g is an element of order q in F_p.

The receiver of (M, r, s) from "person" g^x computes $u = s^{-1}\, SHA(M) \bmod q$ and $v = s^{-1}\, r \bmod q$ and then checks that $((g^u)(g^x)^y \bmod p)$ equals r. If it doesn't, then, by elementary number theory, something definitely went wrong. If it does, then, according to NIST, you can safely assume that message M came from the presumably unique individual who knows the discrete logarithm of g^x. Below we discuss the variant of DSA, using ECC, i.e., Elliptic Curve Digital Signature Algorithm (ECDSA).

Elliptic Curve Digital Signature Algorithm (ECDSA): Elliptic Curve Diffie-Hellman (ECDH)[59] and Elliptic Curve Digital Signature Algorithm (ECDSA)[78] are the Elliptic Curve counterparts of the Diffie-Hellman key exchange and Digital Signature Algorithm, respectively.

The Elliptic Curve Digital Signature Algorithm (ECDSA) works in much the same way as DSA. The only significant difference between ECDSA and DSA is in the generation of r. It requires an elliptic curve E over a finite field F_p with a point P of (large prime) order q, all of which is public. Each user is identified by a scalar multiple, xP, where, x is a (secret) number between 1 and q. To sign a message, the possessor of x again secretly picks a number k between 1 and q and appends a pair of numbers r and s. The number s is exactly as it is for DSA: k^{-1}(SHA (M) +xr) mod q. But this time r is obtained by computing the point kP on the elliptic curve E over F_p: If $kP = (k_1, k_2)$ is the result of that computation, then $r = k_1 \bmod q$. To obtain a security level similar to that of the DSA, the parameter n should have about 160 bits. If this is the case, then DSA and ECDSA signatures have the same bitlength (320 bits).

Verifying an elliptic curve signature is also similar to the verification process for DSA: The receiver computes u and v as before, but then computes the point up+v(xP) on the elliptic curve E over F_p. If the x-coordinate of this point is not equal to r (mod q), the signature is rejected.

The computationally expensive steps in ECDSA are the scalar multiplications kP, uP and v(xP). (The computation of xP is done only once, by the owner of x.) These computations aren't hard in the P-versus-NP sense. They're just tedious and time-consuming. But the special algebraic nature of elliptic curves presents some cost-cutting opportunities. Scalar multiplication, which a naive approach does by repeated doubling and straightforward addition (for example, 9P = 6P+2P+P, where, 2P = P+P and 6P = 2P+2P+2P), is, from the perspective of abstract algebra, a group endomorphism that is, a mapping of the group E_p into itself. As such, it can sometimes be re-expressed in terms of other endomorphisms that, while theoretically fancier, are easier to compute, much as an appropriate change of basis can turn an ugly matrix multiplication problem into a computational breeze.

Instead of using system-wide parameters, we could fix the underlying finite field F_q for all entities and let each entity select its own elliptic curve E and point $P \in (F_q)$. In this case, the defining equation for E, the point P and the order n of P must also be included in the entity's public-key. If the underlying field F_q is fixed, then hardware or software can be built to optimize computations in that field. At the same time, there are an enormous number of choices of elliptic curves E over the fixed F_q.

The mechanics of ECDSA algorithm: We now describe ECDSA as an example of a digital signature algorithm. ECDSA is the elliptic curve analogue of the DSA, which is most famous signature protocol. This protocol needs not only the elliptic curve operations, such as scalar multiplication, field multiplication and field inverse multiplication, but also big integer multiplication, big integer inverse multiplication, modular operation and SHA-1, which is the 160 bit hash function.

In digital signature algorithm, the sender sends a message with the sender's own unique signature and the receiver validates the received signature. Instead of providing a signature for the entire message, the message is first shortened to a fixed length by a hash function, then a short signature that is valid over the whole message is generated.

In the ECDSA, Alice (A) generates the signature with her secret-key and Bob (B) verifies the signature with A's public-key. The algorithm below is the ECDSA protocol which A signs the message m and B verifies A's signature.

The key generation

In ECC, the system parameters such as the prime p, elliptic curve E, base point G(x, y) and order n of the point G need to be shared between the sender and the receiver.

Key generation: Alice (A)

1. Select a random integer s, where $1 \le s \le n-1$,
2. Compute Q = sG the public-key of the sender.
3. A's public-key Q; A's private-key is s.

The signature is generated as follows:

Signature generation: (A)

1. Hash the message and obtain the hash value f = hash(m) = SHA-1(m).
2. Generate a random number u, where $1 \le u \le n-1$
3. Compute $R = uG = (x_R, y_R)$ and $h = d^{-1} \bmod n$.
4. If c = 0 then go to step 1.
5. Compute $d = u^{-1}(f+sc) \bmod n$
6. If d = 0 then go to step 1.
7. Output (c, d) as a signature of m.
8. Sends the message m and the signature (c, d) for the message m, to B.

The signature is validated as follows:

The receiver (B) receives the message m and the signature (c, d). Then, the receiver B performs the following procedure to validate the signature:

1. Verify that c and d are integers in [1, n-1].
2. Hash the message and obtain the hash value f = SHA-1(m).
3. Compute $h = d^{-1} \bmod n$.
4. Compute $h_1 = fh \bmod n$ and $h_2 = ch \bmod n$.
5. Compute $P = h_1G+h_2Q = (x_p, y_p)$ and $c' = x_p \bmod n$.

6. If c = c′, then the signature is valid. Otherwise, it is invalid and rejects the signature.

As can be seen in the above algorithms, we need various operations for implementing the ECDSA protocol. However, we are only concerned with the elliptic curve operations, because these operations dominate the execution time in the protocol schemes. In this protocol, there are two operations, which are: the scalar multiplication in signature generation step 3 and the signature validation step 5 and, the point addition in the signature validation step 5.

How secure is ECDSA? Like all cryptosystems based on the unproved assumptions of complexity theory, that question will have no definitive answer unless some breakthrough shows the answer to be "No." In 1998, Silverman, an elliptic curve expert at Brown University, sent shock waves through the elliptic curve crypto community when he proposed a new method for attacking the xP problem. Silverman's "xedni" algorithm ("xedni" is "index" spelled backward) uses sophisticated techniques in algebraic geometry to pry loose the x. Fortunately (for cryptographers), the xedni algorithm turns out to require exponential time.

Which elliptic can be chosen?

Supersingular elliptic curve [trace 0]: An elliptic E/F_p, $q = p^m$, is called supersingular if $Tr(\varphi_q)$ is divisible by p, Eq. (21). Lets take the case where, $q = p^m$. In this case we have $Tr(\varphi_q) = 0$ or $\#E(F_p) = p + 1$. A remarkable construction in order to reduce the ECDLP to a DLP in extension of F_p[79]. This construction is based on advance algebraic geometry for elliptic curves. It appears that the degree of the extension is 2 for supersingular curves over F_p. The ECDLP can be solved in subexponential time. To avoid this attack larger fields have to be used, in order to guarantee the same cryptographic security, which reduces the speed.

The crucial step in the construction of MOV is based on the use of Weil pairing. Let P, Q and R be points of order n, nP = nQ = nR = O. Let $k \in ¥$ such that $E[n] \subset E(F_{p^k})$. The Weil pairing e_n is a map which sends a pair of points to a n-th power of unity. This n-th power of unity can be embedded in F_{p^k}. Therefore, the Weil pairing can be defined by: $e_n : e[n] \times [n] \rightarrow F_{p^k}$. The Weil pairing has the following properties:

1. $e_n(P, O) = 1$
2. $e_n(P, P) = 1$
3. $e_n(P, Q) \cdot e_n(Q, P) = 1$
4. $e_n(P, R) \cdot e_n(Q, R) = e_n (P+Q,R)$

Suppose that Q = mP. Then for arbitrary points R we have $e_n(P, R)^m = e_n(Q, R)$. The problem to find an m such that $x^m = y$ over F_{p^k}, for $x = e_n (P, R)$ and $y = e_n(Q, R)$. That is, in the case of supersingular curves over F_p the ECDLP can be reduced to a "normal" discrete logarithm over F_{p^2}. Menezes, Okamoto and Vanstone were able to deduce that in case of supersingular curves over F_q, the ECDLP can be reduced to DLP over a field of maximal F_{p^6}.

Anomalous elliptic curves [Trace 1]: Elliptic curves E/F_p with $Tr(\varphi_q)$ are called anomalous. If q = p, then it could be a good idea to chose an anomalous elliptic curve for cryptographic purposes, because the attack of Pohlig-Hellman cannot be applied. At the end of the last century Frey and Rück, Semaev, Satoh and Araki. Smart simplified the isomorphism between $E(F_p)$ and the additive group of F_p[80]. In the last group the discrete logarithm problem can be solved easily. It occurs that an explicit isomorphism $\Psi: E(F_p) \rightarrow Z/pZ = F_p^+$ can be found easily:

$\Psi: E(F_p) \rightarrow F_p$

$P \rightarrow \Psi(P)$

If we would like to calculate a value of m for two points P and Q such that Q = mP, then this can be done by $m = \Psi(Q)/\Psi(P)$.

Elliptic curves of trace 2: Work of G. Frey, M. Müller and H.G. Rück (1998), based work of J. Tate and S. Lichtenbaum, has interesting applications in the ECDLP[80]. This result is in fact an extension on the work of Menezes, Okamoto and Vanstone for supersingular curves. If p is a prime such that P′ = (p-1)/2 is prime as well and E/F_p is an elliptic curve with $Tr(F_p) = 2$, i.e., $\#E(F_p) = p-1$, then the ECDLP can be solved directly and reduced to the DLP in F_p in polynomial ($\log_m$) time. The used definitions and techniques are based on advanced algebraic geometry[81].

ECC in practice

ECC domain parameter: Elliptic curve parameters over the finite F_q or F_{2^m} can be described by one septuple:

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

- q: the prime p or 2^n defines the field and at the same time decides the curve from;
- FR: field representation, a method to express the elements in the field (polynomial basis or normal basis for F_{2^m}, Montgomery for F_n);
- a,b: the curve coefficients, depending on the security requirement . $N = \#E(F_q)$ is divisible by n;
- G: the base point: $G = (x_G, y_G)$, one element in $E(F_q)$, which has the largest order n;
- n: the order of G, large prime;
- h: $\#E(F_q)/n$.

ECC system setup: There are several choices to be made in implementing ECC system[82]:

- Selecting a finite field F_q (and a field representing e.g., np = prime or $p = 2^k$.
- Selecting an appropriate elliptic curve E/F_p (selecting a and b) (plus a point P of order q): random curve vs. a special curve etc.
- Selecting the algorithm for implementing the elliptic curve representation: windows method in affine versus projective coordinates etc.
- Selecting elliptic curve cryptography protocol (e.g., ECDH, ECDSA …) for the task.

Note: Some protocols require additional steps; e.g., ElGamal and others require message embedding: $m \rightarrow P_m \in E(F_q)$.

There are some other requirements on the parameters to defend some sorts of attacks:

- $\#E(F_q)$ 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 a anomalous curves;
- n doesn't divide q^k-1 for $1 \leq k \leq 30$, to resist Weil-Paring and Tate-Paring attacks;
- 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).

Selection criteria

- Security of ECDLP[62]
- Implementation requirements:
- Speed, storage, power consumption.
- Optimization of field/EC operations of protocol.
- Platform dependence: speed and performance of primitives e.g., on a Pentium PC, the time for multiplication is only small multiple of that for addition.
- Standards Compatibility: Public Key Infrastructure (PKI), Wassenaar Arrangement (Export).

Remark: Vanstone (Field Institute Conference, 1999) emphasizes that all these selection criteria must be considered simultaneously[11].

Security of ECC: Not every elliptic curve offers strong security properties and for some curves the ECDLP may be solved efficiently. Since a poor choice of the curve can compromise security, standards organizations like NIST and SECG have published a set of recommended curves[82,83] with well understood security properties. The use of these curves is also recommended as a means of facilitating interoperability between different implementations of a security protocol.

Theoretical security: The strength of every cryptographic algorithm relies on the best methods that are known to solve the mathematical problem, the algorithm is based upon. For all these problems, there are special-purpose algorithms that solve the problem quickly for certain special instances. However, these cases are easy to identify, therefore it is possible to avoid them in an implementation and we will not consider them.

Security of all the systems lays on one or more of their parameters. It means that security level of the system increases by increasing the size of this parameter(s). The primary security parameter of RSA is the modulus N. The DLP has two primary security parameters-order of the underlying group and order of its subgroup - primes p and q. The primary security parameter of ECDLP is the order n of a generator of an additive group of elliptic curve points.

For the ECDLP, the best known general-purpose attack is the Pollard-ρ algorithm, whose running time is fully exponential, i.e.,:

$$T[n] = O(\sqrt{n})$$

Thus, to solve an ECDLP instance for an elliptic curve having a 2k-bit parameter n takes about the same time as the exhaustive search through all k-bit keys of a symmetric cryptosystem (assuming that there is no better attack on the symmetric cryptosystem than the exhaustive search).

Pollard-ρ algorithm is also the basis for the best known general-purpose attack on the subgroup DLP. Therefore, the size of q in the DLP systems should have about the same size as the order n in ECDLP to get the same level of security[84].

The best known general-purpose attack on the IFP (i.e., on N) as well as on the discrete logarithm problem (i.e. on p) are based on the General Number Field Sieve Method which runs in sub-exponential time, i.e.

$$T[x] = O(\exp(c+o(1) \cdot \ln^{1/3} x \cdot \ln^{2/3} \ln x))$$

where, $c = (64/9)^{1/3} \approx 2.923$, x equals to N for the IFP and p for the DLP and the O(1) term goes to zero as x goes to infinity[1]. This means that, to obtain the same level of security, it has to hold

$$|p| = |N| \qquad |q| = |n| \qquad |n| = |2k|$$

and

$$|n| = 2(c + o(1)) \cdot \ln^{-2/3} 2 \cdot |N|^{1/3} \cdot \ln^{2/3}(|N| \ln 2)$$

where, $|\cdot|$ is the length of a parameter in bits and k is key size of a symmetric cryptosystem. The following table (Table 4) compares the equivalent security level for some commonly considered key sizes[84].

Table 4: Computationally equivalent key size

Symmetric schemes (key size in bits)	RSA (n in bits)	DLP (p in bits)	DLP (q in bits)	ECC (n in bits)
56	512	512	112	112
80	1024	1024	160	160
112	2048	2048	224	224
128	3072	3072	256	256
192	7680	7680	384	384
256	15360	15360	512	512

General attacks on elliptic cryptosystem: Over the years mathematicians have made several attempts to tests the robustness of encryption algorithms, but no successful general-purpose algorithms exist. While several examples of special-purpose algorithms have already been mentioned, which exploit special features of the factored numbers, the running times of a general-purpose algorithm only depend on the sizes of the keys[38,39,61].

The cryptographic strength of elliptic curve encryption lies in the difficulty for a cryptanalyst to determine the secret random number k from kP and P itself which in itself is linked to DLP. Hence, the security of an ECC depends on the difficulty of solving the discrete logarithm problem over elliptic curves. The general methods of solving this problem are known. The fastest method to solve this problem (known as the elliptic curve logarithm problem) is the Pollard ρ factorization method[24,85]. Although this is an exponential time algorithm with complexity $O(\sqrt{n})$, making it slow overall, the Rho method has equivalent or better running time than any other elliptic curve attacks. This method has proven its flexibility by solving any instance of the ECDLP, regardless of the type, representation or order of the finite field over which the curve is defined. The Rho method has the additional selling points of ease of implementation, negligible storage requirements and the ability to be sped up through parallelization in software or hardware.

Until recently, however, the best attacks on elliptic curve logarithm problems were the general method and applicable to any group. The methods have a running time of about a constant times the square root of r on average, which is much slower than specialized attacks on certain types of groups. The lack of specialized attacks means that shorter key sizes for elliptic curve cryptosystems give the same security as larger keys in cryptosystems that are based on discrete logarithm problem. However, for certain elliptic curves, Menezes, Okamoto and Vanstone have been able to reduce the elliptic logarithm problem to a discrete logarithm problem[86]. It is possible that algorithm development in this area will change the security of elliptic curve discrete logarithm cryptosystems to be equivalent to that of general discrete logarithm cryptosystems; this is an open research problem.

The square root method is the most general attacking method for the discrete logarithm problem and its computation time is proportional to the exponent of half the key length; that is, the computation time varies exponentially with respect to the key length. A public-key cryptosystem is regarded as being very secure against an attack if the attack takes an exponential amount of time with respect to the key length. From this criterion, we can say that ECCs are very secure against the square root method.

Another attack technique is the Pohlig-Hellman (SPH) method[38], which factors the order of a curve into small primes and solves the Discrete Logarithm Problem (DLP) as a combination of discrete logarithms for small numbers. The SPH method is effective only when the order of the curve is expressed as a product of small primes. Otherwise, the computation time is equivalent to that of the square root method. Therefore, for an ECC, if we select the order of the elliptic curve to be a prime or nearly a prime whose factors include a large prime, the computation time needed to break the ECC will vary exponentially. Therefore, a high level of security can be achieved.

Now we will compare the security of ECCs with that of RSA. The security of RSA lies in the difficulty of factoring large numbers[2]. The number field sieve[86] is the most effective known method for factoring large numbers and it takes a sub-exponential amount of computation time with respect to the key length to do the task. Therefore, the best-known attack against RSA takes a sub-exponential amount of computation time with respect to the key length. An attacking method with sub-exponential time/key-length relationship takes less time than one with an exponential relationship (and more time than a method with polynomial relationship). This is why the securities of 1024-bit RSA and 160-bit ECC are equivalent.

However, since it is impossible to guarantee that a certain problem is infeasible to solve, most people trust RSA as mathematicians can't seem to find an algorithm to efficiently break it, despite the fact that they have been working hard on it, for a long time now (RSA was first developed in 1977)[5]. On the other hand, ECC (developed in 1985) has attracted a lot of criticism as some mathematicians believe that, ECDLP is not infeasible, its just that, this area of study is still very immature as enough research has not yet been done, therefore its just a matter of time before ECC can be broken.

Concept of security evaluation: In general, the security of a cryptosystem is evaluated by the amount of time needed to break it. "Breaking a cryptosystems" means finding the private-key used to encrypt a message. The method used to break a cryptosystem is called the "attacking or brute-

force method". Normally, the time needed to brute-force a practical cryptosystem is never actually obtained, because a cryptosystem that can be broken in reasonable amount of time would not be considered for practical purposes.

Instead, the amount of time needed to break a cryptosystem is a theoretical estimate of the average time needed to break a cryptosystems by a given attacking method. If there are multiple attacking methods, the time required by the most efficient method would be taken as the average time needed to break the cryptosystem. A security evaluation based on the theoretical estimate of average time is valid for the general case, but it is clear that such an evaluation is invalid for special cases.

The computational complexity for breaking the elliptic curve cryptosystem, using the Pollard ρ method, is 3.8×10^{10} MIPS year (i.e., millions of instructions per second times the required number of years) for an elliptic curve key size of only 150-bits[28]. For comparison, the fastest method to break RSA, using the General Number Field Sieve Method (GNFSM) to factor the composite integer n into the two primes p and q, requires 2.0×10^{8} MIPS year for a 768 bit RSA key and 3.0×10^{11} MIPS year with a RSA key of length 1024-bits.

If the RSA key length is increased to 2048-bits, the GNFSM will need 3.0×10^{20} MIPS year to factor n whereas increasing the elliptic curve key length to only 234-bits will impose a computational complexity of 1.6×10^{28} MIPS year (still with the Pollard ρ method).

Evaluation by special attack: The security evaluation of a general attack against an ECC given above is based on the average computation time. However, we cannot evaluate special cases with this evaluation. In fact, special attacks were found using the special characteristics of special elliptic curves[24,79]. The special characteristics are determined by the order of the elliptic curve. These special attacks are much stronger than the square-root method.

Generating a secure curve parameter: Because of the threat of special attacks, it is essential to obtain the parameters of elliptic curves that meet the following basic requirements:

- The order of the curve must be prime or nearly prime.
- The curve must be immune to special attacks.

Both of these requirements concern the order of the curve, #E. That is, the security of an elliptic curve depends primarily on its order. Therefore, to make an ECC secure, we must first find curves which have an order satisfying the above requirements. At present, two method[44,87] are proposed to find good curves:

1) Generate a curve randomly, count its order and select a curve satisfying the criteria.
2) Select an order satisfying the requirements and then generate a curve of the selected order.

For method (1), the Schoof algorithm is used. Theoretically, the computation time for the Schoof algorithm is polynomial with respect to the key length. However, in practice it takes a long time to calculate if we implement it directly in the present computational environment.

For method (2), the complex multiplication algorithm (CM algorithm)[87] is used. The order of computation for the general CM algorithm is exponential with respect to the key length, so it is hard to calculate. Therefore, in practice the order of the curve is selected to have special characteristics so that it can be calculated efficiently. However, as described in the previous section, there is the possibility of attacks using the special characteristics. We think it is doubtful that curves generated by method (2) above will be safe in the future. We therefore recommend the implementation of method (1) coupled with improved Schoof's algorithm.

Merit of ECCs: The merit of ECCs is that compared with RSA cryptosystems they can provide the same security level with a shorter key length. Because of this mathematical property, ECCs are faster and require less hardware than RSA. The security of an ECC, however, depends not only on the length of the key, but also on the elliptic curve parameters. In general, it takes a long time to generate secure parameters. So, faster parameter generation is important for practical implementation of an ECC.

Most Internet security protocols (e.g., SSL[16,88], IPsec) employ a public-key cryptosystem to derive symmetric-keys and then use fast symmetric-key algorithms to ensure confidentiality, integrity and source authentication of bulk data. RSA is the most commonly used public-key cryptosystem today. The security of a system is only as good as that of its weakest component; for this reason, the work factor needed to break a symmetric-key must match that needed to break the public-key cryptosystem used for the key establishment. Due to expected advances in cryptanalysis and increases in computing power available to an adversary, both symmetric and public-key sizes must grow over time to offer acceptable security for a fixed protection lifespan.

Table 5: Proposed the following minimum key sizes (in bits)[56]

		SDL		EC
Year	RSA	q	p	wo (w)*
2000	952	123	952	132 (132)
2005	1149	131	1149	139 (147)
2025	2174	158	2174	169 (202)
2050	4047	193	1447	206 (272)

*without (with) cryptanalytic progress

Table 5[30] shows this expected key-size growth for various symmetric and public-key cryptosystems.

As shown in Table 5, the Elliptic Curve Cryptosystem (ECC) offers the highest strength per bit of any known public-key cryptosystem today. ECC not only uses smaller keys for equivalent strength compared to traditional public-key cryptosystems like RSA, the key size disparity grows as security needs increase. This makes it especially attractive in power, bandwidth and computational savings.

However, the true benefit and performance impact of any cryptosystem is closely tied to how it is used within a security protocol. In particular, it is imperative that expected performance improvements at the protocol level be carefully weighed against the usual costs associated with deploying any new technology.

Applications and implementation of ECC: Thanks to the reduced memory and processing time required to implement ECC, it has several significant advantages over either of its public-key predecessors. Certain distinct implementations, which take advantage of the benefits of ECC, are:

General improvements: Many server and financial applications process large volumes of transaction data (or Web server requests), for which public-key cryptography is required. Even with significant improvements in processing power, server applications can become seriously burdened by conventional public-key operations. ECC can increase efficiency of these operations by orders of magnitude in most cases.

Constrained channels: Are defined as any bottleneck where computational power is limited at one or both ends. Extreme examples of these are wireless communications, where there is a speed/memory tradeoff. However, ECC allows for smaller keys and faster processing times. This significantly improves the protocol execution speeds and reduces power consumption (this is significant in portable/wireless communication devices such as cell-phones and pagers).

The elliptic curve digital signature algorithm: The ECDSA is an algorithm for implementing a digital signature algorithm for the Elliptic Curve Cryptosystem. ECDSA is similar to the DSA, which is the current standard in Digital Signature creation:

- Both use the ElGamal signature scheme and signing equation.
- In both algorithms, the system parameters are difficult to generate, while the private and public keys are relatively simple.
- Both methods use the SHA-1 hash function.

The two approaches do have some significant differences:

- The private key d and the per-signature value k in ECDSA are defined to be statistically unique and unpredictable rather than merely random with DSA. This distinction is important, as with ECDSA it is possible to filter the values to ensure that there are no repeats.
- By a technique known as point compression it is possible to represent a point on an elliptic curve by one field element and one additional bit, rather than two field elements. Public keys can then be represented as 161 bit strings, which is in the order of 25% smaller than other asymmetric algorithms.
- DSA performs an optional bounds check to ensure that a conforming system which did not perform a bounds check cannot generate a signature which it cannot verify. This operation is performed explicitly in ECDSA to avoid any inadvertent application errors.
- ECDSA has the capability of a deterministic primality test, as opposed to the probabilistic test of DSA. This allows for very high security standards.
- In ECDSA, it is possible for a single party to generate their own personal system parameters, while DSA still uses trusted third party.
- It has been demonstrated that the hash function used by DSA is in fact not as reliable as that used with ECDSA. If the adversary can select system parameters, this weakness can be capitalized on. This weakness can be countered by selecting system parameters stipulated in the X9.30 standard.

Automatic teller machines (ATM): Public-key encryption has been in place in the ATM network for a number of years. Storing every user PIN at each terminal is impractical, so security between the terminal and the banking network is imperative. Confidentiality of this PIN

is certainly required while it is in transit and when a large amount of money is requested; the information requires data integrity. This is to combat tampering attacks involving the alteration of the amount requested. An ECC implemented in this environment would be beneficial for certain reasons:

- The reduced key sizes and computations should significantly reduce transaction times, making it more convenient for the user.
- By using ECDSA, the integrity of the data could be more readily guaranteed than the current method of DSA.

Smart cards and cellular phone networks: Phone cards and more generally Smart Cards, are in widespread use throughout North America and Europe. In most instances, important data is stored on the card, such as the amount of money available on it. The security of this data should be protected to avoid fraudulent use and thus free access to the telephone network. It is imperative that the data stored on the card be guaranteed to be genuine and thus some form of security is necessary.

Smart Cards possess a small processor and are capable of storing relatively small amounts of data in memory, making them an exciting avenue for development. These include:

- Use of the cards for ID validation.
- Use as credit cards to replace the magnetic strip cards.
- Electronic cash-replacing the coins and notes we carry with a single card.
- Storage of a patients medical history.

Another area of telephone communications that is vulnerable to fraud is the use of cell phones. It is relatively easy for fraudulent use of the cellular communications network, as the system currently in place can be easily duped. It is also simple to intercept any message sent through the cellular communication network-at present no conversation is free from possible eavesdropping. The Cellular Telecommunications Industry Association estimates that the current generation of cellular phones, without security, enables fraudulent use at a cost of between $1.5 and $2.5 billion each year. Cellular phone providers are only now addressing this issue. The next generation of cellular phones will employ cryptosystems to prevent fraud.

The application of cryptosystems to these examples is obviously important and due to the restrictive processing power and memory resources, ECC is more practical than the alternative public-key cryptosystems.

CONCLUSION

Public-key encryption can be used to eliminate problems involved with conventional encryption; these include key distribution and digital signatures. It however has not managed to be as widely accepted as conventional encryption because it introduces a lot of overheads. Therefore, it is very important to find ways to reduce the overheads yet not sacrificing on other aspects of security so that the desirability in public-key can be exploited.

We have described ECC, which is a promising candidate for the next-generation public-key cryptosystem. Although ECC's security has not been completely evaluated, it is expected to come into widespread use in various fields in the future because of its compactness and high performance when it is hardware-implemented. It is concluded that the reliability, maturity and difficulty of a mathematical problem are very important; the more the difficulty the shorter the keys become hence overheads are eliminated.

After comparing the RSA and ECC cryptosystems, the ECC has proved to involve much less overheads when compared to the RSA. The ECC has been shown to have many advantages due to its ability to provide the same level of security as RSA yet using shorter keys. However, its disadvantage which may even hide its attractiveness is its lack of maturity, as mathematicians believe that enough research has not yet been done in the ECDLP.

Also, we believe that even though both systems are valid, the RSA is better than ECC for now, as it is more reliable and its security can be trusted more. However the future of ECC looks brighter than that of RSA as today's applications (e-commerce, cellular telephones, pagers, smart cards and so on) cannot afford the overheads introduced by RSA. The applications cannot afford to sacrifice their bandwidth for the overheads. Finally, both systems can be considered good cryptosystems considering the low success rates associated with attacking them.

REFERENCES

1. Menezes, A., P. Van Oorschot and S. Vanstone, 1997. Handbook of Applied Cryptography. CRC Press. Inc. Florida.
2. Rabah, K., 2003. A review of RSA and Public-key cryptosystems. To Appear in the Botswana J. Technol. (In Press).
3. Diffie, W. and M.E. Hellman, 1966. New directions in cryptography. IEEE Trans. Inform. Theory, 6: 644-654.

4. Davis, R.M., 1978. The data security standard in perspective. Computer security and the data encryption standard. National Bureau of Standards Special Publication, pp: 500-27.
5. Rivest, R., A. Shamir and L.M. Adleman, 1983. Cryptographic Communications Systems and Method. U.S. Patent 4,404,829,20.
6. Rabin, M.O., 1979. Digital signature and public-key functions as intractable as factorization, MIT laboratory of computer science, Technical report, MIT/LCS/TR-212, Jan., http://www.kisa.or.kr/technology/sub1/Rabin.htm
7. National Institute of Standards and Technology, 1993. Digital Signature Standard, FIPS PUB., 186.
8. ElGamal, T., 1985. A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inform. Theory, 31: 469-472.
9. Merkle, R.C., 1978. Secure communication over insecure channels. Communications of the ACM., 4: 294-299.
10. Bruce, S., 1993. Applied Cryptography. John Wiley and Sons New York.
11. Rivest, R., A. Shamir and L. Adleman, 1978. A method for obtaining digital signatures and public key cryptosystems. Comm. ACM., 21: 120-126.
12. Koblitz, N., 1987. Elliptic curve cryptosystems. Mathematics of Computation, 48: 203-209.
13. Miller, S., 1986. Use of Elliptic Curves in Cryptography. In CRYPTO '85, pp: 417-426.
14. Robshaw, M.J.B. and Yiqun Lisa Yin, 1997. Elliptic curve cryptosystems. An RSA Laboratories Technical Note, pp: 10.
15. Silverman, J.H., 1985. The Arithmetic of Elliptic Curves. Springer-Verlag.
16. Frier, A., P. Karlton and P. Kocher, 1996. The SSL3.0 Protocol Version 3.0. http://wp.netscape.com/eng/ssl3/
17. Silverman, J.H. and J. Tate, 1992. Rational Points on Elliptic Curves. Springer-Verlag.
18. Rubin, K. and A. Silverberg, 1994. Wiles' proof of fermat's last theorem. Bull. Amer. Math. Soc., 31: 15-38.
19. Hellegouarch, Y., 1971. Points d'ordre fini sur les courbes elliptiques. In Acad. Sci. Paris S'er. A-B 273: A540-A543.
20. Anonymous, 1985. National institute of standards and technology (formerly national bureau of standards), FIPS PUB 113: Computer Data Authentication, May 30.
21. Lidl, R. and H. Niederreiter, 1994. Introduction to Finite Fields and Their Applications. Cambridge University Press.
22. Koblitz, N., 1993. Introduction to Elliptic Curves and Modular Forms. Springer-Verlag.
23. Gordon, D.M., 1998. A survey of fast exponentiation methods. J. Algorithms, 27: 129-146.
24. Pollard, J., 1974. Theorems on factorization and primality testing. In Proceedings of the Cambridge Philosophical Society, 76: 521-528.
25. Reisel, H., 1987. Prime Numbers and Computer Methods for Factorization, 2nd Edn., Birkhauser.
26. Satoh, T. and K. Araki, 1998. Fermat quotients and the polynomial time discrete log algorithms for anomalous elliptic curve. Comm. Math. Univ. Sancti, Pauli, 47: 81-92.
27. Menezes, A.J., 1993. Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publishers.
28. tallings, W., 1999. Cryptography and Network Security: Principles and Practice. 2nd Edn. Prentice-Hall, Upper Saddle River, New-Jersey.
29. Odlyzko, A., 1984. Discrete Logarithms in Finite Fields and Their Cryptographic Significance. In Advances in Cryptology Eurocrypt'84, Springer-Verlag, pp: 224-314.
30. Koblitz, N., 1987. Elliptic curve cryptosystems. Mathematics of Computation, 48: 203-209.
31. Rosing, M., 1998. Implementing Elliptic Curve Cryptography. Manning Publications Co.
32. Bernd, M. and M. Volker, 1996. A public key cryptosystem based on elliptic curves over z/nz equivalent to factoring. Eurocrypt, pp: 33-48.
33. Itoh, T. and S. Tsujii, 1988. A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. Info. and Comput., 78: 171-177.
34. Grossschädl, J., 2001. A bit-serial unified multiplier architecture for finite fields GF(p) and GF(2^m). In Workshop on Cryptographic Hardware and Embedded Systems (CHES '01), pp: 40.
35. Hasse-Weil's Theorem, 1934. (Weil's conjecture, proved by Helmut Hasse in1934.) http://www.opengroup.com/mabooks/354/3540564896.shtml
36. Orlando, G. and C. Paar, 1999. A super-serial Galois field multiplier for FPGAs and its application to public key algorithms. In Proceedings of the IEEE Symposium on Field-programmable Custom Computing Machines (FCCM'99), pp: 232-239.
37. Pohlig, S.C. and M.E. Hellman, 1978. An improved algorithm for computing logarithms over GP(p) and its cryptographic significance. IEEE Trans. Inform. Theory, 24: 106-110.
38. Morain, F., 1998. Building Cyclic Elliptic Curves Modulo Large Primes. Advances in Cryptology-EUROCRYPT'91, LNCS, Springer-Verlag, 547: 328-336.

39. Shanks, D., 1970. Class number, a theory of factorization and genera, Proc. Symposium of Pure Math. Amer. Math. Soc., 20: 415-440.
40. Mestre, J.F., 1986. Formules explicites at minoration de conducteurs de variete algebriques, Composition Math., 58: 209-232.
41. Arjen K. Lenstra and Hendrik W. Lenstra, Jr., 1990. Algorithm in Number Theory. Handbook of Theoretical Computer Science, J. van Leeuwen (Ed.), MIT Press, Cambridge, Massachusetts, pp: 673-715.
42. François, M., 1991. Building Cyclic Elliptic Curve Modulo Large Primes. LNCS 547, Advances in Cryptology-Eurocrypt'91, Brighton, United Kingdom, Donald W. Davies (Ed.), Springer-Verlag, pp: 328-336.
43. Georg-Johann, L. and Z. Horst-Günter, 1994. Constructing elliptic curves with given group order over large finite fields, LNCS 877, Algorithmic Number Theory. The 1st International Symposium, ANTS-I, Ithaca, Springer-Verlag, pp: 250-263.
44. Schoof, R., 1998. Elliptic curves over finite fields and computation of square roots mod p. Math. Comp., 44: 483-494.
45. Joux, A. and R. Lercier, 2001. Chinese and match, An alternative to Atkin's "Match and sort" method used in the SEA algorithm. Math. Comp., 70: 827-836.
46. Pollard, J. M., 1978. Monte Carlo methods for index computation (mod p), Math. Comp., 32: 918-924.
47. Paul, C. van O. and M.J. Wiener, 1994. Parallel collision search with applications to hash functions and discrete logarithms. The 2nd ACM Conference on Computer and Communications Security, ACM Press, New York, November, pp: 210-218.
48. Paul, C. van O. and M.J. Wiener, 1999. Parallel collision search with cryptanalytic applications. J. Cryptol., 12: 1-28.
49. Shoup, V., 1997. Lower Bounds for Discrete Logarithms and Related Problems, LNCS 1233. Advances in Cryptology-Eurocrypt'97, Konstanz, Germany, Walter Fumy (Ed.), Springer-Verlag, pp: 256-266.
50. Silverman, R.D. and J. Stapleton, 1997. Contribution to ANSI X9F1 working group. http://www-306.ibm.com/security/standard/st_activity0897.shtml.
51. Flassenberg, R. and P. Sachar, 1997. Sieving in function fields. The ECDLP workshop '97, University of Waterloo, Ontario, Canada.
52. Silverman, J.H., 1999. The xedni calculus and the elliptic curve discrete logarithm problem. Technical Report CORR 99-05, University of Waterloo, Ontario, Canada.
53. Silverman, J.H. and J. Suzuki, 1999. Elliptic Curve Discrete Logarithms and the Index Calculus, LNCS 1514. Advances in Cryptology-Asiacrypt '98, Beijing, China, Kazuo Ohta and Dingyi Pei (Eds.), Springer-Verlag.
54. Koyama, K., U.M. Maurer, T. Okamoto and S.A. Vanstone, 1991. New Public-key Schemes Based on Elliptic Curves over the Ring Zn, CRYPTO'91 Abstracts, Santa Barbara, CA., August 11-15, pp: 6-1.
55. Demytko, N., 1994. A New Elliptic Curve Based Analogue of RSA. In T. Hellsesth, (Ed.), Advances in Cryptology-Eurocrpt'93, Springer-Verlag, New York, pp: 40-49.
56. Agnew, G.B., R.C. Mullin, I.M. Onyszchuk and S.A. Vanstone, 1991. An implementation for a fast public-key cryptosystem. J. Cryptol., 3: 63-79.
57. Schnorr, C., 1991. Efficient Signature generation by smart cards. J. Cryptol., pp: 161-174.
58. Nyberg, K. and Rueppel, 1996. Message recover for signature schemes based on the discrete logarithm problem. Designs, Codes and Cryptography, 7: 61-81.
59. Koblitz, N., 1993. Introduction to Elliptic Curves and Modular Forms. Springer-Verlag.
60. Rabin, M.O., 1979. Digital signature and public-key functions as intractable as factorization. MIT Laboratory of Computer Science, Technical report, MIT/LCS/TR-212, Jan.
61. Lenstra, H.W., 1987. Factoring integers with elliptic curves. Ann. Math., 126, pp: 649-673.
62. Semav, A., 1998. Evaluation of discrete logarithm on some elliptic curve, Math. Comp., 67: 353-356.
63. Menezes, A. and D. Johnson, 1998. Elliptic curve DSA: An Enhanced DSA. www.certicom.ca.
64. Diffie, W., P. van O. and M. Wiener, 1992. Authentication and authenticated key exchanges, Designs, Codes and Cryptography, 2: 107-125.
65. Diffie, W. and M. E. Hellman, 1976. Multi-user cryptographic techniques. Proceedings of AFIPS National Computer Conference, pp: 109-112.
66. Schroeppel, R., H. Orman, S. O'Mally and O. Spatscheck, 1995. Fast Key Exchange with Elliptic Curve Systems. In CRYPTO '95, pp: 43-56.
67. Hellman, M. and J. Reyneri, 1983. Fast computation of discrete logarithms in GF(q). Advances in Cryptology: Proceedings of Crypto '82. Plenum Press.
68. Lim, C. and P. Lee, 1997. A key recovery attack on discrete log-based schemes using a prime order subgroup. Advances in Cryptology-Crypto '97, Lecture Notes in Computer Science, 1294: 249-263.

69. Bernd, M. and M. Volker, 1996. A Public Key Cryptosystem Based on Elliptic Curves over Z/nZ Equivalent to Factoring. LNCS 1070. Advances in Cryptology-Eurocrypt '96, Saragossa, Spain, Ueli M. Maurer (Ed.), Springer-Verlag, pp: 49-59.
70. Boneh, D., 1998. The decision Diffie-Hellman problem, LNCS 1423, algorithmic number theory. The 3rd international symposium, ANTS-III, Portland, Oregon, Joe P. Buhler (Ed.), June, pp: 48-63.
71. Boneh, D. and R. Venkatesan, 1996. Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes, LNCS 1109, Advances in Cryptology-Crypto '96, Santa Barbara, California, Neal Koblitz (Ed.), Springer-Verlag, pp: 129-142.
72. Boneh, D. and I.E. Shparlinski, 2001. On the Unpredictability of Bits of the Elliptic Curve Diffie-Hellman scheme, LNCS 2139. Advances in Cryptology-Crypto 2001, Santa Barbara, California, Joe Kilian (Ed.), Springer-Verlag, pp: 201.
73. He, J. and T. Kiesier, 1994. Enhancing the security of ElGamal's signature scheme. IEE Proceedings of Computing and Digital Technique, 141, 4, July, pp: 249-252.
74. Massey, J.L. and J.K. Omura, 1986. U.S. Patent No. 4,567,600. Method and apparatus for maintaining the privacy of digital messages conveyed by public transmission, 28 January.
75. Menezes, A.J. and S.A. Vanstone, 1990. The implementation of elliptic curve cryptosystems, LNCS 453. Advances in Cryptology-Auscrypt '90, Sydney, Australia, Josef Pieprzyk and Jennifer Seberry (Eds.), Springer-Verlag, pp: 2-13.
76. Menezes, A.J. and S.A. Vanstone, 1993. Elliptic curve cryptosystems and their implementation, J. Cryptol., 6: 209-224.
77. Basham, L., Johnson and D.T. Polk, 1999. Representation of elliptic curve digital signature algorithm (ECDSA) Keys and Signatures in Internet X.509 Public-Key Infrastructure Certificates. Internet Draft, June, Available at: http://www.ietf.org.
78. ANSI X9.62, 1999. The elliptic curve digital signature algorithm (ECDSA). American Bankers Association. http://csrc.nist.gov/cryptval/dss.htm
79. Menezaes, A.J., T. Okamoto and S.A. Vanstone, 1993. Reducing elliptic curve logarithms to a finite field. IEEE Trans. Inform. Theory, 39: 1639-1646.
80. Coppersmith, D., A.M. Odlyzko and R. Schroeppel, 1986. Discrete logarithms in GF(p), Algorithmica, 1: 1-15.
81. Koblitz., N., 1987. A Course in Number Theory and Cryptography. Springer-Verlag.
82. Standards for Efficient Cryptography Group, EC 1: Elliptic Curve Cryptography, version 1.0, 2000, Available at: http://www.secg.org.
83. Standards for Efficient Cryptography Group, SEC 2: Recommended Elliptic Curve Domain Parameters, version 1.0, 2000. Available at: http://www.secg.org
84. Johnson, D.B., 1999. ECC, future resiliency and high security systems. http://www.certicom.com/resources/w_papers/w_papers.html, March.
85. Atkin, O. and F. Morain, 1993. Elliptic curves and primality proving. Math. Comp., 61: 29-68.
86. Daniel, M.G., 1993. Discrete logarithm in GF(p) using the number field sieve. SIAM J. Discrete Math., 6: 124-138.
87. oblitz, N., 1992. CM-curves with good cryptographic properties. In Advances in Cryptology: Proceedings of Crypto '91, volume 576, Springer-Verlag, pp: 279-287.
88. Dierks, T. and C. Allen, 1999. The TLS Protocol-Version 1.0, IETF RFC 2246. http://www.javvin.com/protocolTLS.html and http://www.faqs.org/rfcs/rfc2246.html