http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 4 (2): 132-139, 2005
ISSN 1812-5638

Implementation of Elliptic Curve Diffie-Hellman and EC Encryption Schemes

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

Abstract: In recent years the rapid deployment of applications like online banking, stock trading and corporate remote access, have seen an explosive growth in the amount of sensitive data exchanged over the internet. Moreover, these internet hosts increasingly are battery-powered, wireless, handheld devices with strict memory, CPU, latency and bandwidth constraints. Given these trends, there is a clear need for efficient, scalable security mechanisms and protocols that operate well in both wired and wireless environments. To date elliptic curve cryptography is gaining wide acceptance as an alternative to the conventional cryptosystems (DES, RSA, AES, etc.) which tend to be power hungry. Elliptic curve ciphers require less computational power, memory and communication bandwidth giving it a clear edge over the traditional crypto-algorithms. This study describes the basic design principle of Elliptic Curve Crypto (ECC), EC discrete logarithm problem, ECDH key agreement and encryption protocols.

Key words: Cryptography, data security, secure communication, wireless, ECC, ECDA, DH, ElGamal, RSA

INTRODUCTION

The rapid growth of information technology that has resulted in significant advances in cryptography to protect the integrity and confidentiality of data is astounding. The historical cryptosystem has been a symmetric-key approach, in which both communicating partners share a common key and a level of trust is required to ensure that neither party divulges the key[1,2]. Considered by some to be the only major abstract advance in encryption techniques, Diffie and Hellman[3] proposed Public Key Encryption in 1976 and has remained a very popular protocol for strong authentication of entities[4]. By this method each user of the network has a personalized private key and a public key. The public key is distributed to all members of the network, while only the user holds the private key. This implies that the user is the only person able to read any messages encrypted with his public key. It is designed to be computationally intractable to calculate a private-key from its associated public-key. With public key cryptography, more sophisticated cryptographic tools such as tokens (e.g., smart cards), digital signatures and certificates are used to provide the full scope of cryptographic security services.

However, it is important to note that traditional cryptographic algorithms like DES[1], DLP[5], RSA[6] etc. are not particularly efficient in small form factor, low-power, resource constrained devices, as they require memory intensive power hungry big integer computation co-processor to complete the calculations in a timely manner. Adding such a co-processor significantly raises the cost of manufacture, rendering many devices impractical. The cost of producing a smart card, for example, is increased by as much as 400% when an additional processor is required[7]. For embedded systems or telecommunications applications characterized by extremely high volumes and a wide variety of devices, many of which have limited computing resources and wireless, the trend has been towards alternate cryptographic algorithms.

One technology in particular, known as Elliptic Curve Cryptography (ECC), has become the cryptography of choice for mobile computing and communications devices due to its size and efficiency benefits. Koblitz[8] and Miller[9] independently proposed the Elliptic Curve Cryptosystem (ECC), a method of utilizing a Discrete Logarithm problem over the points on an elliptic curve. By their proposal, ECC could be used to provide both digital signatures and an encryption scheme. Over the past decade, ECC and later ECDLP (Elliptic Curve Discrete Logarithm Problem) received considerable attention from mathematicians around the world[10], but to-date no significant breakthroughs have been made in determining weaknesses in the algorithm and to-date has weathered unipteen mathematical attacks[11,12]. Elliptic curve systems have thereby come to be accepted today as the most viable public-key technology for high-security applications.

The ECC provides higher strength-per-bit than any other current public-key cryptosystems[12]. If you compare elliptic curves to RSA and DLP you find many

advantages: to obtain the same security e.g., the use of smaller fields for elliptic curves. Therefore, elliptic curves can be implemented easier and faster. Moreover, because of its higher strength-per-bit, ECCs is being increasingly used in practical applications (e.g. IC card and mobile devices) instead of RSA, which is the most used public-key cryptosystems today. They are also most suitable for constrained environments such as those in which smart cards and personal wireless devices are typically deployed. Elliptic curve ciphers are today commonly found in smart cards, personal digital assistants (PDAs), pagers and mobile phones. The Elliptic Curve Cryptosystems are also used for implementing protocols such as ECDSA digital signature scheme[13,14], Diffie-Hellman key exchange scheme[3], EC ElGamal Encryption/Decryption scheme[15] etc. In this study we analyze the elliptic curve operations and design procedure involving ECDLP, ECDH key exchange agreement and EC encryption protocols.

ECC ARITHMETIC OVER GALOIS FIELD

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. Let an elliptic curve group over the Galois Field $E_p(a, b)$ where, $p>3$ and is prime, be the set of solutions or points $P = (x, y)$ such that $(x, y \in E_p(a,b))$ that satisfy the equation: $y^2 = x^3+ax+b \pmod p$ 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 constants a and b are non negative integers smaller than the prime number p and must satisfy the condition: $4a^3+27 b^2 \neq 0 \pmod p$. 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 $E_p(a, b)$. So there will be a lot of points modulo p. In fact, the general theory says that there will be about p points (x, y) with error bounded by $O(\sqrt{p})$.

Construction of an elliptic curve over F_p: Let the prime number p =23 and consider an elliptic curve E: $y^2 = x^3+x+4$ mod 23 defined over F_{23}, with the constants a = 1 and b=4, which have been checked to satisfy that E is indeed an elliptic curve. We then determine the quadratic residues Q_{23} from the reduced set of residue $Z_{23} = \{1,2,3,.....,21,22\}$, which is given by $Q_{23} = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$. Which we use to determine the values of $E_{23}(1,4)$, i.e.,:

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

$$\begin{Bmatrix} (4, 7) & (4, 16) & (7, 3) \\ (9, 12) & (10, 5) & (10, 18) \\ (14, 5) & (14, 18) & (15, 6) \\ (18, 14) & (22, 5) & (22, 18) \end{Bmatrix}$$

Group Order-Let E be the elliptic curve order 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 $\# E(F_q) = q+1-t$ where $|t| \le 2\sqrt{p}$, $\# E(F_q)$ is called the order of an elliptic curve E and t is called the trace of E[16]. In other words, the order of an elliptic curve $E(F_q)$ is roughly equal to the size q of the underlying field[17]. However, since the number of points in our elliptic curve is small, we can use the naïve approach to determine $\#E(F_q)$ using scalar multiplication/addition approach.

In the development and implementation of elliptic curve cryptography we are interested in the method for computing an equation of the form kP where, k is an integer in the range of [1, n-1], n is the order of the elliptic curve E and $P = (x_p, y_p) \in E(F_q)$ is a non zero point on a given elliptic curve E. Here, P is usually a fixed point that generates a large, prime subgroup of $E(F_q)$including the point at O, or P is an arbitrary point in such a subgroup. This system of computation is known as scalar multiplication and is the heart of ECC scheme.

Point addition: Scalar multiplication i.e., the computation of kP, where k is a random integer and P is an elliptic curve generation point, can be defined as the combination of additions of two points on an elliptic curve. 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. Its properties, computation and uses will be, therefore, the core of ECC implementation. The addition of two points on an elliptic curve is defined in order that the addition results will be another point on the curve as presented in Algorithm 1 and Fig. 1.

Algorithm 1: Point addition equation

Input: $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$
Output: $P_1+P_2 = P_3 = (x_3, y_3)$

1. If $P_1 = P_2$: $x_3 = \lambda^2+\lambda+a$, $y_3 = x^2_1+(\lambda+1)x_3$ where , $\lambda = x_1+y_1/x_1$ (Point doubling)
2. Else if $P_1 \neq P_2$: $x_3 = \lambda^2+\lambda+x_1+x_2+a$, $y_3 = \lambda(x_1+x_3)+x_3+y_1$ where, $\lambda = (y_1+y_2)/(x_1+x_2)$ (Point addition)
3. Return: (x_3, y_3)

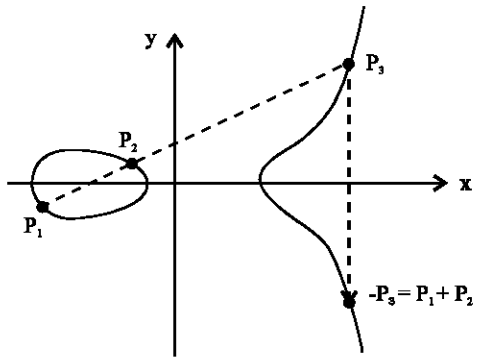

Fig. 1: Elliptic curve

In either case, when $P_1 = P_2$ (doubling) and $P \neq P_2$ (point addition), major operations are field multiplication and field inversion. (Squaring and field addition are enough ignorable because of its less computation time.) From these formulas of Algorithm 1, we can determine the number of field operations required for each kind of elliptic curve operation. We see that an addition step usually requires eight additions, two multiplications, one squaring, three reductions mod f(x) and one inversion. A Doubling step usually requires four additions, two multiplications, two squaring, four reductions mod f(x) and one inversion. A Negation step requires one addition. The important contributors to the run time are the multiplications and inversions[17]. Just as modular exponentiation determines the efficiency of RSA cryptographic systems[4], scalar multiplication dominates the execution time of ECC systems. In all the protocols that are fundamental implementation of ECC, say ECDH, ECDSA, ECAES etc., the most time consuming part of the computations are scalar multiplications. Elliptic curves have some properties that allow optimization of scalar multiplications.

Implementation of multiplication/addition over an elliptic curve group modulo p: Let us consider an equation of the form Q = kP. For a positive integer k we let [k] denote the multiplication-by-k map from the curve to itself. This map takes a point P to P+P+.....+P (k summands). The notation [k] is extended to $k \leq 0$ by defining [0]P = O and [-k]P = -([k]P). So for instance, [2]P = P+P, [3]P = P+P+P and [-3]P = -(P+P+P). Furthermore, for a given point P on an elliptic curve E, 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. These scalar multiples form a subgroup of points<P>. Here,<P> is the finite cyclic group<P> = {P, 2P, 3P,.....,nP}, with order n.

To test the algorithm, let $P = (7,3) \in E_{23}(1,4)$. Then $2P = (x_3, y_3)$ is equal to: $2P = P + P = (x_1, y_1) + (x_1, y_1) = (22, 18)$. Next, test addition of two different points on the curve, i.e., $Q = (4,16) \in E_{23}(1, 4)$ and $R = (14,18) \in E_{23}(1, 4)$. Then $Q+R = (x_3, y_3) = (17,9)$, which we can observe also to be on the curve. However, for real implementation of ECC, we need to know the order of the elliptic curve group.

Let us now implement the scalar multiplication to form a subgroup of points<P>, where,<P> is the finite cyclic group<P> = {P, 2P, 3P,.....nP}, with order n, by following the same additive rules and a generator point P. For example, let $P = (7,3) \in E_{23}(1,4)$ be a generator point which we use through repeated addition of point to generate all the point on the curve (Table 1).

Cyclic elliptic curve-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 $E(F_{23})$. The order n of a point $P \neq O$ on an elliptic curve is a positive integer such that nP = Q and $mP \neq O$ for any integer m such that $1 \leq m < n$. 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 $N = \#E(F_q)$ is a prime number, then the group is cyclic and obviously all points except the point at infinity O are of order N and which is the case looking from Table 1.

ECC PROTOCOLS

Key establishment schemes: Key establishment is the process by which two (or more) entities establish a shared cryptographic secret key and are essential for distribution of keys in today's communication systems for use with symmetric and asymmetric cryptography. Essentially, two methods are used to establish cryptographic keying material between parties: key agreement and key transport. With a key agreement scheme, all parties contribute to the derived keying material with information that allows each party to derive the shared keying material. With key transport schemes, the sender determines the key to be transported, wraps (i.e., encrypts) the key and sends the wrapped key to the receiver, who then unwraps (i.e., decrypts) the key.

For the scheme which involves symmetric-key only system, the communicating partners have to manually establish a shared symmetric-key to be used as the key-wrapping key between the two parties. The keying material is wrapped using a NIST-approved key-wrapping algorithm (such as the AES key wrap algorithm). The public-key based key agreement schemes can be transformed into a key transport scheme using an

Table 1: Point addition/scalar point multiplicative values of P

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

$kP = P_k$, is that $29P = P_0$ = [O]etc

approved key-wrapping scheme, as recommended in NIST SP 800-56. An example is S/MIME, where key agreement is combined with key wrapping to achieve the effect of key transport. This is useful because it allows the efficient encryption of large emails to multiple recipients. The efficiency is that the email content is encrypted just once, with the content encryption key encrypted (wrapped) multiple times, once for each recipient.

Elliptic Curve Discrete Logarithm Problem (ECDLP): Recall that the core of elliptic curve arithmetic is an operation called scalar point multiplication, which computes Q = kP. (For example, 11P can be expressed as 11P = (2*((2*(2*P))+P))+P). The problem of calculating k from a given points P and Q is called the discrete logarithm problem over the elliptic curve (ECDLP). 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 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 prime finite field F_q.

Implementation of ECDLP algorithm: As a simple example, let's use an overly small value of k and our earlier elliptic curve i.e., $E_p(a, b) = E_{23}(1, 4)$ (Table 1). Let's choose a point $(1, 12) \in E_{23}(1, 4)$ and try to determine k such that kP = Q. For example, if Q= (1, 12) and P=(7, 3), then 5P = Q so k = 5 is the solution to the discrete logarithm problem. One can also compute Q = kP = 1P+1P+1P+1P+1P = (1, 12) = 5P. Here k=5. Alternatively, given Q = kP = (1, 12), we can continuously subtract P from kP until we get O. But if the curve is of a large prime order then it is impossible to use this naïve approach for our computation. Take, for example, an elliptic curve $E_p(a, b)$ of a large prime number with:

p = 6,277,101,735,386,680,763,835,789,423,207,666,416, 083,908,700,390,324,961,279

containing a large group of points exactly N points:

N = 6,277,101,735,386,680,763,835,789,423,337,720,473,986 773,608,255,189,015,329

with

k = 6,708,050,311,399,110,513,517,527,207,693,060,456,300 217,054,473

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 $\#E(F_q) = n.h$, where, h is a small integer. 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 (as indicated above) 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.

KEY AGREEMENT PROTOCOL

EC Diffie-Hellman key agreement protocol: The Diffie-Hellman key agreement protocol is the basic public-key cryptosystem proposed for computing key sharing agreement for both private and public key protocols. ECDH is the elliptic curve analog of the traditional Diffie-Hellman key agreement algorithm[1,3,4]. The Diffie-Hellman method requires no prior contact between the two parties. Each party generates a dynamic, or ephemeral, public key and private key. They exchange their public keys. Each party then combines its private key with the other party's public key to compute the shared secret[18].

As an example of ECDH key agreement scheme let us consider a home-banking subscriber, entity A (Alice), setting up a secure communication channel with her Bank entity (B). Alice and the Bank first agree to use a specific curve, field size and type of mathematics. Alice generates a public key and a private key; she sends the public key to her bank. Independently, her bank generates a public key and a private key; the bank's public key is sent to Alice. Alice combines her private key and the bank's public key to form a shared secret. Her bank combines its private key and Alice's public key to arrive at the same shared secret key. The shared secret may now be used to generate a shared key for encrypting and decrypting the communication sessions, as shown in Algorithm 2. We can see that we just need scalar multiplication in order to implement the ECDH protocol.

Algorithm 2: Diffie-Hellman protocol

1. A and B each chose random private key k_A and k_B, respectively.
2. A and B each calculate $A = k_AP = (x_A, y_A)$ and $B = k_BP = (x_B, y_B)$; and send them to opposite side.
3. A and B both compute the shared secret: $Q = k_A(k_BP) = k_B(k_AP) = kP = (x_Q, x_Q)$;s hared key is $k = k_Ak_B$.

- Note that in step 2 we can also compute hk_AP and hk_BP, which can resist the attack on small subgroup. where h is a co-factor defined in P1363

Implementation of ECDH key agreement protocol: 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 $k_A = 12$ and computes her public-key (Table 1): $Q_A = k_AP = 12P = (13, 11)$. Similarly, Bank (B) chooses the secret-key $k_B = 23$ and computes its public-key: $Q_B = k_BP = 23P = (0, 2)$.

Thus, their common secret-key $S_{AB} = k_Ak_B$. Alice computes: $k_AQ_B = 12(23P) = 15P = (17, 9)$ and the Bank computes: $k_BQ_A = 23(12P) = 15P = (17, 9)$. Such that: $S_{AB} = k_AQ_B = k_BQ_A = 15P = (17, 9)$.

An alternative form of ECDH is the Elliptic curve decision Diffie-Hellman problem (ECDDHP) Boneh[19]. 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), (sP) and (mP), the ECDDHP is to decide whether m = kl (modulo the order of point P). The ECDDHP is not harder than the ECDHP. Boneh, Venkatesan[20], Boneh and Shparlinski[21] discussed many issues on security of the ECDHP and related schemes.

Key agreement and setup scheme: The ECDH Key Agreement Scheme as specified in ANSI X9.63 and IEEE P1363 In ECKAS-DH1 (the Elliptic Curve Key Agreement Scheme, Diffie-Hellman 1), each party combines its own private key with the other party's public key to calculate a shared secret key which can then be used as the key for a symmetric encryption algorithm such as AES. Other (public or private) information known to both parties may be used as key derivation parameters to ensure that a different secret key is generated in every session. This key agreement scheme is described in more detail in section 9.2 of the IEEEP1363 standard.

Pure DH or ECDH applications are, however, susceptible to impersonation or "man in the middle" attack, whereby an adversary establishes digital facades between two parties in order to obtain private information. For example, Alice may be setting up a secure session with someone impersonating her bank. Because the private and public keys are generated on the fly, there is no way to prove you have a secure session with the intended party unless you add a method for user authentication. Advanced key exchange protocols such as the Menezes-Qu-Vanstone (MQV) introduce mutual strong authentication which allows both parties to confidently identify each other before exchanging sensitive information. MQV is currently deployed through ECC systems[4,19].

THE MECHANICS OF 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 message 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 $E_p(a, b)$ and the generator G are made public.

Assume that the Bank and Alice intends to communicate. Each user select a private key and use it to compute their public-key. For example, Alice (A) selects a random integer $k_A<n$ as her private-key and computes her public-key P_A as: $P_A = k_AG$. To encrypt the message P_M to the Bank (B); Alice uses her private-key and the Bank's public-key P_B to compute the ciphertext pair of points P_C:

$$P_C = [(k_AG), (P_M+k_AP_B)]$$

After receiving the ciphertext pair of points, P_C, the Bank multiplies the first point, (k_AG) with its private-key, k_B and then adds the result to the second point in the ciphertext pair of points, $(P_M+k_AP_B)$:

$$(P_M+k_AP_B)-[k_B(kG)] = (P_M+k_Ak_BG)-[k_B(k_AG)] = P_M$$

which is the plaintext point, corresponding to the plaintext message M. Only the Bank, knowing the private-key k_B,

can remove $k_B(k_AG)$ from the second point of the ciphertext pair of point, i.e., $(P_M+k_AP_B)$ and hence retrieve the plaintext information P_M.

Implementation of Elliptic Curve Encryption Scheme (ECES): Since $\#E(F_{23}) = 29$, which is prime. $E(F_{23})$ is cyclic and any point other than O is a generator of $E(F_{23})$. For example, $G \equiv P = (7,3)$ is a generator point such that the multiples kG of the generator point G (for $1 \le k \le 29$), are as shown in Table 1.

If Alice wants to send to Bank the message M which is encoded as the plaintext point $P_M = (10, 18) \in E_{23}(1, 4)$. She must use the Bank's public-key to encrypt it. Suppose that the Bank's secret-key is $k_B = 17$, then its public-key will be: $P_B = k_BG = 17(7, 3) = 17G = (13, 12)$

Alice uses her secret-key $k_A = 25$ and Bank's public-key $P_B = (13, 12)$ to encrypt the message point into the ciphertext pair of points: $P_C = [(k_AG), (P_M+k_AP_B)] = [(4, 16), (22, 5)]$

Upon receiving the ciphertext pair of points, $P_C = [(4, 16), (22, 5)]$, the Bank uses its private-key, k_B, to compute the plaintext point, P_M, as follows:

$$(P_M+k_AP_B)-[k_B(k_AG)] = 27G-19G = 8G (10, 18)$$

and which maps the plaintext point $P_M = (10, 18)$ back into the original plaintext message M.

Practical Application of ECES: For this part let's get real and simulate real application. Let an elliptic curve group over the Galois Field $E_p(a, b)$ where, $p>3$ and is prime, be the set of solutions or points $P = (x, y)$ such that $(x, y \in E_p(a, b))$ that satisfy the equation: $y^2 = x^3+ax+b(\text{mod } p)$ for $0 \le x < p$ together with the extra point O called the point at infinity. The parameters of the elliptic curve are as follows:

a = 3176890812513255034763174764138276932727469 55927
b = 7905289660787875871812057202571853543210065 1934
p = 8596310237942882237669478944689739620749856 8951
n = 7859631023794288223766930248817149576126861 57429 \\number of points #E
G = (x, y) \\ generator point

Where:

x = 7715072162626498261706482685655798899077692 54176
y = 9015 7510246556628525279459266514995562533196655

k_A = 6708050311399110513517527207693060456300217054473 \\Alice's priv. key
k_B = 7860050311399110513517527207693060456300217054 473 \\Bank's priv. key
k_M = 2355050311399110513517527207693060456300217054473 \\Message enc. key

This is playing Big Boys game so we need a number cruncher software to do similar computation done above. Here we will content ourselves with PARI a Free, open source, number cruncher[22].

Here the curve $E(F_p)$ is cyclic and any point other than O is a generator of all points on curve. For example, G = (x, y) is a generator point such that the multiples kG of the generator point G (for $1 \le k \le n$), which is too huge to show here.

If the Bank wants to send to Alice the message M (Append. A), which is encoded as the plaintext point

$P_M = (x_M, y_M) = (28512811040597236886373218071701660114124205872 1, 9487390262232107223294077342398725955 8219668492)$

The Bank must use Alice's public-key to encrypt it. Suppose that Alice's secret-key is k_A, then her public-key will be:

$P_A = k_AG = (90517135491294082980111512034183665522 537517216, 3004291515310424027770375706419816028 45272146478)$

The Bank selects a random number k_B and Alice's public-key P_A to encrypt the message point into the ciphertext pair of points:

$P_C = [(k_BG), (P_M+k_BP_A)]$
$= [(5506053616725573161131116154825881464349 84456674, 5982067025845974155376420114956358149258942 03618), (1723908357770572622250717936439507557160 65775457, 7426170520788598780501120202393221074990282 98971)]$

Upon receiving the ciphertext pair of points, P_C, Alice uses her private-key, k_A, to compute the plaintext message point, P_M, as follows:

Table 2: NIST guidelines for public-key sizes with equivalent security levels

Security (bits)	Symmetric encryption algorithms	Minimum size (bits) of public keys		
		DSA/DH	RSA	ECC
80	Skipjack	1024	1024	160
112	3DES	2048	2048	224
128	AES-128	3072	3072	256
192	AES-192	7680	7680	384
256	AES-256	15360	15360	512

$(P_M+k_AP_B)-[k_B(k_AG)] = P_M$

$= (2851281104059723688637321807$
$1701660114124205872 1,$
$9487390262232107223294077342 3$
$98725955821966849 2)$

and which maps the plaintext point $P_M = (x_M, y_M)$ back into the original plaintext message M.

Elliptic Curve Integrated Encryption Scheme (ECIES): ECIES combines elliptic curve asymmetric encryption and the AES symmetric encryption algorithm with the SHA-1 hash algorithm to provide an easy to use encryption scheme with message authentication support. An ECIES ciphertext object (Q, C, T) consisting of EC public key Q, encrypted message C and authentication tag T is generated from a message M and the recipient's EC public key W. The recipient decrypts this ciphertext with their EC private key and an exception is thrown if the authentication tag is invalid. This encryption scheme is described in more detail in section 11.3 of the IEEE P1363a draft standard[23].

Following in the footsteps of DES[1]; ECC in conjunction with advance symmetric algorithm, AES[24], has already been incorporated into a number of key international standards, including ANSI X9.63, IEEE Std 1363-2000, IETF RFC 3278, ISO 15946-3 and NIST SP 800-56[2]. Adoption into global standards will assist in pushing ECC into wider commercial usage. Table 2 compares the equivalent security level for some commonly considered cryptographic key sizes[19].

Relative public-key sizes: So what does this mean in practice? NIST has recommended that 128-bit protection is necessary to achieve relatively lasting security (to the year 2036 and beyond). This means moving from 3DES to AES. To avoid compromising the security of the system, NIST's FIPS 140-2 standard indicates that keys for symmetric ciphers such as AES must be matched in strength by public-key algorithms such as RSA and ECC. For example, a 128-bit AES key demands an RSA key size of 3,072-bits for equivalent security, however, for the same strength the ECC key size is only 256-bits. As you can observe from Table 2, while ECC key sizes scale linearly, RSA does not. The result is that the gap between systems grows as the key sizes increase. This is especially relevant to implementations of AES where at 256-bit security you need an RSA key size of 15,360-bits compared to 512-bits for ECC. This will have a significant impact on a communication system as the relative computational performance advantage of ECC versus RSA is not indicated by the key sizes but by the cube of the key sizes. The difference becomes even more dramatic as the greater increase in RSA key sizes leads to an even greater increase in computational cost. So going from 1024-bit RSA key to 3072-bit RSA key requires about 27 times (3^3) as much computation while ECC would only increase the computational cost by just over 4 times (1.6^3).

CONCLUSION

We have shown that elliptic curve ciphers require less computational power, memory and communication bandwidth giving it a clear edge over the traditional crypto-algorithms. To date elliptic curve cryptography is gaining wide acceptance, especially in wireless and hand-held devices, when compared to the conventional cryptosystems (DES, RSA, AES, etc.) which tend to be power hungry. However, while the performance advantages are impressive with ECC, the data security industry need to ensure that the security system, using elliptic curve algorithm has been studied extensively in the public forum and also specified by major standards worldwide. But we think that elliptic curve cryptography is here today and is without question the next generation of public-key cryptography of choice.

Appendix A: Sample text message from the bank to alice

Miss Alice Johnson
Nairobi, Kenya

Dear Miss Johnson,

This letter is to inform you that we have received US$ 2.0 million as your first installment due for your mortgage down payment and opened mortgage servicing account No. 6688668024 on your behalf and deposited therein said amount. To service your account you will need to use your new password GKGQU78BR53.

Yours very truly,
James M Kavungu
Vice President, Finance
First Finance Bank of Nairobi

REFERENCES

1. Kefa, R., 2004. Data security and cryptographic techniques-A review. PJIT., 3: 106-132.
2. Menezes, A., P. Van Oorschot and S. Vanstone, 1997. Handbook of Applied Cryptography. CRC Press.
3. Diffie, W. and M.E. Hellman, 1976. Multi-user Cryptographic Techniques. Proceedings of AFIPS National Computer Conference, pp: 109-112.
4. Rabah, K., 2004. A Review of RSA and Public-key Cryptosystems. BJT., 13: 1-11.
5. Odlyzko, A., 1984. Discrete Logarithms in Finite Fields Andtheir Cryptographic Significance. In: Advances in Cryptology Eurocrypt'84, Springer-Verlag, pp: 224- 314.
6. Rivest, R.L., A. Shamir and L.M. Adleman, 1978. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM., 21: 120-126.
7. Aigner, H., J. Wolkerstorfer, M. Huetter and H. Bock, 2004. Low-cost ECC Coprocessor for Smartcards. Workshop on Cryptographic Hardware and Embedded Systems CHES 2004, Cambridge (Boston), USA.
8. Koblitz, N., 1986. Elliptic Curve Cryptosystems. Mathematics of Computation, 48: 203-209.
9. Miller, V.S., 1986. Use of elliptic curve in cryptography. Advances in Cryptology-Proceedings of CRYPTO'85, Springer Verlag Lecture Notes in Computer Science, 218: 417-426.
10. ECDLP-I., 1998. A. Semav, Evaluation of discrete logarithm on some elliptic curve. Math. Comp., 67: 353-356.
11. Rabah, K., 2004. Elliptic Curve Cryptography. J. Applied Sci., 5: 604-633.
12. Certicom Research, 1997. The Elliptic Curve Cryptosystem, Certicom.
13. ECDSA-L. Basham, D. Johnson and T. Polk, 1999. Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public-Key Infrastructure Certificates, Internet Draft. http://www.ietf.org.
14. ANSI X9.62, 1999. The Elliptic Curve Digital Signature Algorithm (ECDSA), American Bankers Association.
15. ElGamal, T., 1985. A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Info. Theory, 31: 469-472.
16. Hasse-Weil's Theorem (Weil's conjecture, proved by Helmut Hasse in 1934.)
17. Silverman, J.H., 1985. The Arithmetic of Elliptic Curves. Springer-Verlag.
18. Diffie, W., P. van Oorschot and M. Wiener, 1992. Authentication and authenticated key exchanges. Designs, Codes and Cryptography, 2: 107-125.
19. Dan, B., 1998. The decision Diffie-Hellman problem, LNCS 1423, Algorithmic Number Theory, the 3rd International Symposium, ANTS-III, Portland, Oregon, Joe P. Buhler (Ed.), pp: 48-63.
20. Dan, B. 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.
21. Dan, B. 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
22. PARI, http://pari.math.u-bordeaux.fr/
23. IEEE P1363a: http://grouper.ieee.org/groups/1363/P1363a/
24. U.S. Department of Commerce/National Institute of Standards and Technology. Announcing request for candidate algorithm nominations for the Advanced Encryption Standard (AES). World Wide Web, September 12, 1997. http://csrc.nist.gov/encryption/aes/pre-round1/aes_9709.htm.
25. ANSI X9.42 and X9.63. http://cio.doe.gov/Publications/profile2000/Profile2000_AppendixA.htm
26. ANSI key schemes FIPS 140-2. http://cio.doe.gov/Publications/profile2000/Profile2000_AppendixA.htm