УДК 681.3, 004.3

Rodrigue Elias
Computers Department, Institute of Computer Technology,
Automation and Metrology,
Lviv, Polytechnic National University

BASES OF DIGITAL SIGNATURE USING ELLIPTIC-CURVE CRYPTO PROCESSOR OVER GALOIS FIELD (2^M)

Визначено основи для створення нової послідовної архітектури процесора для виконання додавання і скалярного множення точок еліптичних кривих.

In this study, the procedure is to design a base for new hardware processor to perform elliptic curve addition and scalar multiplication. It will be based on a serial architecture.

Introduction

An obvious application of cryptography is the transformation of information to prevent other from observing its meaning. This is the classical concept of secrecy, wherein we attempt to prevent information from reaching an enemy in a usable form. Secrecy is viewed by many as the central issue in the field of information protection. Secure communication is the most straightforward use of cryptography. Two people may communicate securely by encrypting the messages sent between them. While secure communication has existed for centuries, the key management problem has prevented it from becoming commonplace. Thanks to the development of public-key cryptography, the tools exist to create a large-scale network of people who can communicate securely with one another even if they had never communicated before. Cryptography is widely used. Not only is it used over the Internet, but also it is used in phones, televisions, and a variety of other common household items. Without cryptography, hackers could get into our e-mail, listen in on our phone conversations, or break into our bank/brokerage accounts. Cryptography is not confined to the world of computers. Cryptography is also used in cellular (mobile) phones as a means of authentication; that is, it can be used to verify that a particular phone has the right to bill to a particular phone number. This prevents people from stealing ("cloning"') cellular phone numbers and access codes. Another application is to protect phone calls from spying using voice encryption. Many other uses for cryptography exist at present, and further applications will almost certainly grow as our understanding of information protection increases.

Problem description

The major advantage of properly designed and implemented cryptographic applications is that they are inexpensive to use, expensive to attack, and independent of other factors in the environment. The major problems are in the assurance of these advantages in actual use. Now there are many algorithms and standards for cryptographic applications. The problem is to select some algorithms that are more secured, more confidential, and in a way, faster than the previous designed architectures.

Purpose of Work

In the information age, cryptography has become one of the major methods for protection in all applications. Cryptography allows people to carry over the confidence found in the physical world to the electronic world. It allows people to do business electronically without worries of deceit and deception. In the distant past, cryptography was used to assure only secrecy. Wax seals, signatures, and other physical mechanisms were typically used to assure integrity of the message and authenticity of the sender. When people started doing business online and needed to transfer funds electronically, the applications of cryptography for integrity began to surpass its use for secrecy. Hundreds of thousands of people interact electronically every day, whether it is through e-mail, e-commerce (business conducted over the Internet),

ATM machines, or cellular phones. The constant increase of information transmitted electronically has lead to an increased reliance on cryptography and authentication.

The purpose of this work is to select some algorithms that are more secured, more confidential, and in a way, faster than ones used in the previous designed architectures. These algorithms will be implemented in the next following stages.

Introduction to finite fields

Fields are abstractions of familiar number systems (such as the rational numbers Q, the real numbers R, and the complex numbers C) and their essential properties. They consist of a set F together with two operations, addition (denoted by +) and multiplication (denoted by ·), that satisfy the usual arithmetic properties:

(F, +) is an abelian group with (additive) identity denoted by 0.

(F - {0}, ·) is an abelian group with (multiplicative) identity denoted by 1.

The distributive law holds: (a + b) ·c = a ·c + b ·c for all a, b, c ∈ F.

If the set F is finite, then the field is said to be finite.

Field operations

A field F is equipped with two operations, addition and multiplication. *Subtraction* of field elements is defined in terms of addition: for $a, b \in$ F, $a - b = a + (-b)$ where $-b$ is the unique element in F such that $b + (-b) = 0$ ($-b$ is called the *negative* of b). Similarly, *division* of field elements is defined in terms of multiplication: for $a, b \in$ F with $b = 0$, $a/b = a \cdot b^{-1}$ where b^{-1} is the unique element in F such that $b \cdot b^{-1} = 1$. (b^{-1} is called the *inverse* of b.)

Prime fields

Let p be a prime number. The integers modulo p, consisting of the integers $\{0, 1, 2, \ldots, p-1\}$ with addition and multiplication performed modulo p, is a finite field of order p. We shall denote this field by Fp and call p the *modulus* of Fp. For any integer a, a mod p shall denote the unique integer remainder r, $0 \le r \le p-1$, obtained upon dividing a by p; this operation is called *reduction modulo p*.

Binary fields

Finite fields of order 2^m are called *binary fields* or *characteristic-two finite fields*. One way to construct F(2^m) is to use a *polynomial basis representation*. Here, the elements of F(2^m) are the binary polynomials (polynomials whose coefficients are in the field $F_2 = \{0,1\}$) of degree at most $m-1$:

$$F(2^m) = \{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \cdots + a_2z^2 + a_1z + a_0 : a_i \in \{0,1\}\}.$$

An irreducible binary polynomial *f(z)* of degree *m* is chosen (such a polynomial exists for any *m* and can be efficiently found). Irreducibility of *f(z)* means that *f(z)* cannot be factored as a product of binary polynomials each of degree less than *m*. Addition of field elements is the usual addition of polynomials, with coefficient arithmetic performed modulo 2. Multiplication of field elements is performed modulo the *reduction polynomial f (z)*. For any binary polynomial *a(z)*, *a(z)* mod *f(z)* shall denote the unique remainder polynomial *r(z)* of degree less than *m* obtained upon long division of *a(z)* by *f(z)*; this operation is called *reduction modulo f(z)*.

Elliptic Curves

Cryptographic mechanisms based on elliptic curves depend on arithmetic involving the points of the curve. Curve arithmetic is defined in terms of underlying field operations, the efficiency of which is essential. Efficient curve operations are likewise crucial to performance.

Let E be an elliptic curve defined over the field K. There is a chord-and-tangent rule for adding two points in E(K) to give a third point in E(K). Together with this addition operation, the set of points E(K) forms an abelian group with ∞ serving as its identity. It is this group that is used in the construction of elliptic curve cryptographic systems. The addition rule is best explained geometrically. Let P = (x1, y1) and Q = (x2, y2) be two distinct points on an elliptic curve E. Then the sum R, of P and Q, is defined as follows. First draw a line through P and Q; this line intersects the elliptic curve at a third point. Then R is the reflection of this point about the x-axis. The double R, of P, is defined as follows. First draw the tangent line to the elliptic curve at P. This line intersects the elliptic curve at a second point. Then R is the

reflection of this point about the x-axis. Algebraic formulas for the group law can be derived from the geometric description.

How Digital Signature Technology Works

Digital signatures are created and verified by cryptography, the branch of applied mathematics that concerns itself with transforming messages into seemingly unintelligible forms and back again. Digital signatures use what is known as "*public key cryptography*", which employs an algorithm using two different but mathematically related "*keys*"; one for creating a digital signature or transforming data into a seemingly unintelligible form, and another key for verifying a digital signature or returning the message to its original form. Computer equipment and software utilizing two such keys are often collectively termed an "*asymmetric cryptosystem.*"

Main Standards and Algorithms

ECC Standards

Cryptographic standards are important for two reasons: (1) to facilitate the widespread use of cryptographically sound and well-specified techniques; and (2) to promote interoperability between different implementations. Interoperability is encouraged by completely specifying the steps of the cryptographic schemes and the formats for shared data such as domain parameters, keys and exchanged messages, and by limiting the number of options available to the implementer. This section describes the salient features of selected standards and draft standards that describe elliptic curve mechanisms for signatures, encryption, and key establishment.

American National Standards Institute (ANSI).

The ANSI X9F subcommittee of the ANSI X9 committee develops information security standards for the financial services industry. Two elliptic curve standards have been completed: ANSI X9.62 which specifies the ECDSA, and ANSI X9.63 which specifies numerous elliptic curve key agreement and key transport protocols including STS. The objective of these standards is to achieve a high degree of security and interoperability. The underlying finite field is restricted to being a prime field Fp or a binary field $F(2^m)$. The elements of $F(2^m)$ may be represented using a polynomial basis or a normal basis over F_2.

National Institute of Standards and Technology (NIST).

NIST is a non-regulatory federal agency within the U.S. Commerce Department's Technology Administration. Included in its mission is the development of security-related Federal Information Processing Standards (FIPS) intended for use by U.S. federal government departments. The FIPS standards widely adopted and deployed around the world include the Data Encryption Standard (DES) [1], the Secure Hash Algorithms (SHA-1, SHA-256, SHA-384 and SHA-512: FIPS 180-2 [2]), the Advanced Encryption Standard (AES: FIPS 197 [3]), and Hash-based Message Authentication Code (HMAC: FIPS 198 [4]). FIPS 186-2, also known as the Digital Signature Standard (DSS), specifies the RSA, DSA and ECDSA signature schemes. NIST is in the process of developing a recommendation [5] for elliptic curve key establishment schemes that will include a selection of protocols from ANSI X9.63.

Institute of Electrical and Electronics Engineers (IEEE).

The IEEE P1363 working group is developing a suite of standards for public-key cryptography. The scope of P1363 is very broad and includes schemes based on the intractability of integer factorization, discrete logarithm in finite fields, elliptic curve discrete logarithms, and lattice-based schemes. The 1363-2000 standard includes elliptic curve signature schemes (ECDSA and an elliptic curve analogue of a signature scheme due to Nyberg and Rueppel), and elliptic curve key agreement schemes (ECMQV and variants of elliptic curve Diffie- Ellman (ECDH)). It differs fundamentally from the ANSI standards and FIPS 186-2 in that there are no mandated minimum security requirements and there is an abundance of options. Its primary purpose, therefore, is to serve as a reference for specifications of a variety of cryptographic protocols from which other standards and applications can select. The 1363-2000 standard restricts the underlying finite field to be a prime field F*p* or a binary field F2*m* . The P1363a [7] draft standard is an addendum to 1363-2000. It contains specifications of ECIES and the Pintsov-Vanstone signature scheme providing message recovery, and allows for extension fields F*pm* of odd characteristic including optimal extension fields.

International Organization for Standardization (ISO).

ISO and the International Electro-technical Commission (IEC) jointly develop cryptographic standards within the SC 27 subcommittee. ISO/IEC 15946 is a suite of elliptic curve cryptographic standards that specifies signature schemes (including ECDSA and EC-KCDSA), key establishment schemes (including ECMQV and STS), and digital signature schemes providing message recovery. ISO/IEC 18033-2 provides detailed descriptions and security analyses of various public-key encryption schemes including ECIES-KEM and PSEC-KEM.

Standards for Efficient Cryptography Group (SECG).

SECG is a consortium of companies formed to address potential interoperability problems with cryptographic standards. SEC 1 specifies ECDSA, ECIES, ECDH and ECMQV, and attempts to be compatible with all ANSI, NIST, IEEE and ISO/IEC elliptic curve standards.

New European Schemes for Signatures, Integrity and Encryption (NESSIE).

The NESSIE project was funded by the European Union's Fifth Framework Programme. Its main objective was to assess and select various symmetric-key primitives (block ciphers, stream ciphers, hash functions, message authentication codes) and public-key primitives (public-key encryption, signature and identification schemes). The elliptic curve schemes selected were ECDSA and the key transport protocols PSEC-KEM and ACE-KEM.

Cryptographic Research and Evaluation Committee (CRYPTREC).

The Information-technology Promotion Agency (IPA) in Japan formed the CRYPTREC committee for the purpose of evaluating cryptographic protocols for securing the Japanese government's electronic business. Numerous symmetric-key and public-key primitives are being evaluated, including ECDSA, ECIES, PSEC-KEM and ECDH.

Digital Signature Main Algorithms

RSA Systems and Key Generations.

RSA, named after its inventors Rivest, Shamir and Adleman, was proposed in 1977 shortly after the discovery of public-key cryptography. It consists of selecting key pair (public key & private key), so that the problem is, deriving the private key from the corresponding public key. The public key consists of a pair of integers (n, e) where *"n"* is a product of two randomly generated and secret primes (p & q) of the same bitlength. The encryption exponent *"e"* is an integer satisfying $1 < e < \varphi$ & $\gcd(e,\varphi) = 1$ where $\varphi = (p-1)(q-1)$. The private key *"d"*, called the decryption exponent, is the integer satisfying $1 < d < \varphi$ and $ed \equiv 1 \pmod{\varphi}$.

RSA Encryption.

RSA encryption is used to generate a ciphertext *"c"* such that $c = m^e \bmod n$.

RSA Decryption.

RSA decryption works in the opposite direction. Here, we have to find the plaintext *"m"* from the ciphertext *"c"*. The security relies on the difficulty of computing the plaintext *"m"*.

RSA Signing Scheme: The signer of a message *m* first computes its message digest $h = H(m)$ using a cryptographic hash function *H*, where *h* serves as a short fingerprint of *m*. Then, the signer uses his private key *d* to compute the *e*th root *s* of *h* modulo *n*: $s = h^d \bmod n$. The signer transmits the message *m* and its signature *s* to a verifying party. This party then recomputes the message digest $h = H(m)$, recovers a message digest $h' = s^e \bmod n$ from *s*, and accepts the signature as being valid for *m* provided that $h = h'$. The security relies on the inability of an intruder (who does not know the private key *d*) to compute e^{th} roots modulo *n*.

RSA Signature Generation.

To generate a digital signature, RSA public Key (n, e), RSA private key *"d"*, and a message *"m"* must be generated. Then, two steps will be made:

Compute $h = H(m)$ where *H* is a hash function.

Compute $s = h^d \bmod n$.

Finally, attach the signature *s* with the message *m*.

RSA Signature Verification.

At the reception terminal, the RSA public key *(n, e)*, the message *m*, and the signature must be held. To accept or reject the signature, the following procedure must be applied:

Compute $h = H(m)$ where H is the same hash function.

Compute $h' = s^e \bmod n$.

If $h = h'$ then "Accept the signature", Else "Reject the signature".

Discrete Logarithm Systems.

The first discrete logarithm (DL) system was the key agreement protocol proposed by Diffie and Hellman in 1976. In 1984, ElGamal described DL public-key encryption and signature schemes.

DL Key Generation: In discrete logarithm systems, a key pair *(x, y)* is associated with a set of public domain parameters *(p, q, g)*.

DL Domain Parameter Generation.

The following steps must be applied in order to generate the three public domain parameters *(p, q, g)* based on two input parameters *(l, t)*.

i. Select a t-bit prime q and an l-bit prime p such that q divides $(p - 1)$.

ii. Select an element g of order q:

iii. Select arbitrary $h \in [1, p-1]$ and compute $g = h(p - 1)/q \bmod p$.

iv. If $g = 1$ then go to step iii.

v. Return (p, q, g).

DL key pair generation.

Now, the key pair *(x, y)* will be generated from the public domain parameters *(p, q, g)* using the following part of an algorithm.

i. Select $x \in Z\ [1, q-1]$.

ii. Compute $y = g^x \bmod p$.

iii. Return (y, x).

Discrete Logarithm Encryption Scheme.

Next, the most known encryption and decryption for the El-Gammal public key procedures will be presented.

El-Gamal Encryption.

The inputs will be: the DL domain parameters *(p, q, g)*, the public key *(y)*, and the plaintext $(m) \in [0, p-1]$. After that, two ciphertexts (c_1, c_2) will be generated as followed:

i. Select $k \in R\ [1, q-1]$.

ii. Compute $c_1 = g^k \bmod p$.

iii. Compute $c_2 = m \cdot y^k \bmod p$.

iv. Return (c_1, c_2).

El-Gamal Decryption.

Now, the DL domain parameters *(p, q, g)*, the private key *(x)*, and the Ciphertext (c_1, c_2) will be as inputs, to generate back the plaintext *(m)*.

i. Compute $m = c_1 * c_2^{-x} \bmod p$.

ii. Return (m).

Discrete Logarithm Signature Scheme.

An entity A with private key x signs a message by selecting a random integer k from the interval [1, q-1], and computing $T = gk \bmod p$, $r = T \bmod q$, $s = k^{-1}(h + xr) \bmod q$, where $h = H(m)$ is the message digest. A's signature on m is the pair (r, s). To verify the signature, an entity must check that (r, s) satisfies the previous equation. Since the verifier knows neither A's private key x nor k, this equation cannot be directly verified. Note, this equation is equivalent to $k \equiv s^{-1}(h + xr) \pmod q$.

Raising g to both sides yields the equivalent congruence $T \equiv (ghs)^{-1}(yrs)^{-1} \pmod p$.

The verifier can therefore compute T and then check that $r = T \bmod q$.

DSA Signature Generation.

As inputs, DL domain parameters *(p, q, g)*, private key *(x)*, and the message *(m)*, as outputs, Signature *(r, s)*.

i. Select $k \in R [1, q-1]$.
ii. Compute $T = g^k \bmod p$.
iii. Compute $r = T \bmod q$. If $r = 0$ then go to step i.
iv. Compute $h = H(m)$.
v. Compute $s = k^{-1}(h + xr) \bmod q$. If $s = 0$ then go to step i.
vi. Return (r, s).

DSA Signature Verification

As inputs, DL domain parameters *(p, q, g)*, public key *(y)*, message *(m)*, and the signature *(r, s)*. As output, either accept the signature or reject the signature.

i. Verify that r and s are integers in the interval $[1, q-1]$. If any verification fails
ii. then return ("Reject the signature").
iii. Compute $h = H(m)$.
iv. Compute $w = s^{-1} \bmod q$.
v. Compute $u_1 = hw \bmod q$ and $u_2 = rw \bmod q$.
vi. Compute $T = g^{u1} y^{u2} \bmod p$.
vii. Compute $r' = T \bmod q$.
viii. If $r = r'$ then return ("Accept the signature");
ix. Else return ("Reject the signature").

Elliptic Curve Key Pair Generation.

Let E be an elliptic curve defined over a finite field Fp. Let P be a point in E(Fp), and suppose that P has prime order n. Then the cyclic subgroup of E(Fp) generated by P is:$\langle P \rangle = \{\infty, P, 2P, 3P, \ldots, (n-1)P\}$. The prime p, the equation of the elliptic curve E, and the point P and its order n, are the public domain parameters. A private key is an integer d that is selected uniformly at random from the interval $[1, n-1]$, and the corresponding public key is $Q = dP$.

INPUT: Elliptic curve domain parameters *(p, E, P, n)*.
OUTPUT: Public key Q and private key d.

i. Select $d \in Z [1, n-1]$.
ii. Compute $Q = dP$.
iii. Return (Q, d).

Elliptic Curve Encryption Scheme.

Here are the procedures for the elliptic curve analogue of the basic ElGamal encryption. A plaintext m is first represented as a point M, and then encrypted by adding it to kQ where k is a randomly selected integer, and Q is the intended recipient's public key. The sender transmits the points $C1 = kP$ and $C2 = M + kQ$ to the recipient who uses her private key d to compute $dC1 = d(kP) = k(dP) = kQ$, and thereafter recovers $M = C2 - kQ$. An eavesdropper who wishes to recover M needs to compute kQ.

Basic ElGamal Elliptic Curve Encryption.

INPUT: Elliptic curve domain parameters *(p, E, P, n)*, public key Q, plaintext m.
OUTPUT: Ciphertext (C_1, C_2).

i. 1. Represent the message m as a point M in E(Fp).
ii. 2. Select $k \in R [1, n-1]$.
iii. 3. Compute $C_1 = kP$.
iv. 4. Compute $C_2 = M + kQ$.
v. 5. Return (C_1, C_2).

Basic ElGamal Elliptic Curve Decryption:

INPUT: Domain parameters *(p, E, P, n)*, private key d, ciphertext (C_1, C_2).
OUTPUT: Plaintext m.

i. 1. Compute $M = C_2 - dC_1$, and extract m from M.
ii. 2. Return (m).

Why Elliptic Curve Cryptography?

There are several criteria that need to be considered when selecting a family of public key schemes for a specific application [8, 9]. The principal ones are:

Functionality. Does the public-key family provide the desired capabilities?

Security. What assurances are available that the protocols are secure?

Performance. For the desired level of security, do the protocols meet performance objectives?

The RSA, DL and EC families all provide the basic functionality expected of public-key cryptography—encryption, signatures, and key agreement. Over the years, researchers have developed techniques for designing and proving the security of RSA, DL and EC protocols under reasonable assumptions. The fundamental security issue that remains is the hardness of the underlying mathematical problem that is necessary for the security of all protocols in a public-key family—the integer factorization problem for RSA systems, the discrete logarithm problem for DL systems, and the elliptic curve discrete logarithm problem for EC systems. The perceived hardness of these problems directly impacts performance since it dictates the sizes of the domain and key parameters. That in turn affects the performance of the underlying arithmetic operations.

Performance: The *efficiency* of an algorithm is measured by the scarce resources it consumes. Typically the measure used is time, but other measures such as space and number of processors are also considered. It is reasonable to expect that an algorithm consumes greater resources for larger inputs, and the efficiency of an algorithm is therefore described as a function of the input size. Here, the *size* is defined to be the number of bits needed to represent the input using a reasonable encoding. Expressions for the running time of an algorithm are most useful if they are independent of any particular platform used to implement the algorithm. This is achieved by estimating the number of elementary operations (e.g., bit operations) executed. The (worst-case) *running time* of an algorithm is an upper bound, expressed as a function of the input size, on the number of elementary steps executed by the algorithm. Studies are made to calculate the efficiency of the different algorithms; it was found that EC algorithms have the best efficiency.

Security: Estimates are given for parameter sizes providing comparable levels of security for RSA, DL, and EC systems, under the assumption that the algorithms mentioned above are indeed the best ones that exist for the integer factorization, discrete logarithm, and elliptic curve discrete logarithm problems. If time is the only measure used for the efficiency of an algorithm, then the parameter sizes providing equivalent security levels for RSA, DL and EC systems can be derived using the running times. The parameter sizes, also called *key sizes*, that provide equivalent security levels for RSA, DL and EC systems as an 80-, 112-, 128-, 192- and 256-bit symmetric-key encryption scheme are listed in the table below. By a *security level* of k bits we mean that the best algorithm known for breaking the system takes approximately 2^k steps.

RSA, DL and EC key sizes
for equivalent security levels.

	Security level (bits) [6]				
	80	112	128	192	256
DL parameter q	160	224	256	384	512
EC parameter n					
RSA modulus n	1024	2048	3072	8192	15360
DL modulus p					

Note: bitlengths are given for the DL parameter q and the EC parameter n, and the RSA modulus n and the DL modulus p, respectively.

The comparisons in the table above demonstrate that smaller parameters can be used in elliptic curve cryptography (ECC) than with RSA and DL systems at a given security level. The difference in parameter sizes is especially pronounced for higher security levels. The advantages that can be gained from smaller parameters include speed (faster computations) and smaller keys and certificates. In particular, private-key operations (such as signature generation and decryption) for ECC are many times more efficient than RSA and DL private-key operations. Public-key operations (such as signature verification and encryption) for ECC are many times more efficient than for DL systems. Public-key operations for RSA are expected to be somewhat faster than for ECC if a small encryption exponent is selected for RSA. The advantages offered by ECC are important where processing power, storage, bandwidth, or power consumption is constrained.

Conclusion

In this article, cryptography was introduced for its high importance in transferring digital data between terminals without hijacking. In addition to this, several algorithms have been described. Two out of them have been chosen for implementation at next higher stages. Moreover, a comparison has been made between ECC and other techniques, so ECC has been selected to have the most reliable, efficient, and secured algorithms. In this article, an algorithm will be choosen, *Basic El-Gamal Elliptic Curve Encryption-Decryption Scheme*, and implemented it in the next following stages.

1. I. BIEHL, B. MEYER, AND V. MULLER. Differential fault analysis on elliptic curve crypto-systems. Advances in Cryptology — CRYPTO 2000, 131–146, 2000. 2. FIPS 180-2. Secure Hash Standard. Federal Information Processing Standards Publication 180-2, National Institute of Standards and Technology, 2002. 3. FIPS 197. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, National Institute of Standards and Technology, 2001. 4. FIPS 198. HMAC – Keyed-Hash Message Authentication. Federal Information Processing Standards Publication 198, National Institute of Standards and Technology, 2002. 5. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. NIST Special Publication 800-56: Recommendation on key establishment schemes, Draft 2.0. Available from http: //csrc.nist.gov/CryptoToolkit/tkkeymgmt.html, January 2003. 6. D. Hankerson, A. Menezes, S. Vanstone. Guide to Elliptic Curve Cryptography. Springer, N.Y. 2004. 7. IEEE P1363A. Standard Specifications for Public-Key Cryptography. Additional Techniques, 2003. 8. K. Rabah. Theory and Implementation of Elliptic Curve Cryptography. Journal of Applied Sciences, 2005. Asian Network for Scientific Information. 9. K. Rabah. Theory and Implementation of Data Encryption Standard. Information Technology Journal, 2005.