A Novel Digital Envelope Approach for A Secure E-Commerce Channel

Ramachandran Ganesan[1], Mohan Gobi[1], and Kanniappan Vivekanandan[2]
(Corresponding author: R. Ganesan)

Department of Computer Science and Applications, PSG College of Arts and Science[1]
Civil Aerodrome Post,Coimbatore 641 014, India
BSMED, Bharathiar University, Coimbatore 641 046, India[2]
(Email: rganrao@gmail.com)
(Received July 30, 2009; revised and accepted Oct. 15 & 29, 2009)

Abstract

Data privacy and integrity will be the crucial and significant factors in recent times for trade which will be transacted over the Internet through e-commerce and m-commerce channels. To deal with these anxieties, various security etiquette related to symmetric and asymmetric key types have been framed. Digital Envelope is one of the practices to attain Privacy, Authentication, Integrity maintenance, and Non-Repudiation in e-commerce channels. In this paper, we suggest a software implementation of a digital envelope for a secure e-commerce channel that combines the hashing algorithm of MD5, the symmetric key algorithm of AES and the asymmetric key algorithm of Hyper Elliptic Curve Cryptography (HECC). The result illustrates that HECC is the best alternative asymmetric key technique rather than ECC and RSA in the digital envelope hybrid cryptosystem.

Keywords: Advanced encryption standard (AES), authenticity, hyper-elliptic curve cryptosystems (HECC), message digest version 5 (MD5), non-reputability

1 Introduction

The e-commerce market has grown exponentially over the last decade and has really helped merchants to sell their products and services to a much larger and broader base of customers. On account of the success of e-commerce, overseas and international markets are now a much more a plausible opportunity. The prime requirements for any e-commerce transactions are Privacy, Authentication, Integrity maintenance and Non-Repudiation. All these are achieved through cryptographic techniques [8]. Digital envelope is one such mechanism to achieve the same. Most of the digital envelope employs RSA algorithm to encrypt and decrypt the secret key. However, the RSA itself is vulnerable [10]. Therefore, in this research work we emphasize HECC asymmetric key technique as an alternative for ECC and RSA in the digital envelope.

Cryptographic techniques are broadly classified into symmetric key cryptographic techniques (DES, TDES, AES) and asymmetric key cryptographic techniques (RSA, ECC, HECC). In this paper, the implementation details of a digital envelope for a secure e-commerce channel using the AES and HEC Cryptosystem is presented. The reason behind for the adoption of HECC in this approach is that, for the minimal key length, HECC provides more security than ECC and RSA. HECC requires 80 bit key length compared to 1024 bit key length of RSA and 160 bit key length of ECC to provide the same level of security.

In Section 2, the basic of a digital envelope and AES algorithm are highlighted. Hyper-Elliptic Curve Cryptosystem is explicated in Section 3. Details on hashing and message digest are presented in Section 4. Design details of the digital envelope using AES and HECC and its implementation details are demonstrated in Section 5. Results are analyzed in Sections 6 & 7. Finally, Section 8 is a conclusion.

2 Digital Envelope and AES

According to RSA Labs, "The digital envelope consists of a message encrypted using secret-key cryptography and an encrypted secret key. Digital envelopes usually use public-key cryptography to encrypt the secret key." Details of digital envelope can be had from http://www.rsa.com/. An additional enhancement to the digital envelope is integrity maintenance using hash functions. Figure 1 shows how a digital envelope is created.

Advanced Encryption Standard (AES) is an approved symmetric key cryptographic algorithm that is a block cipher. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits.

The algorithm consists of four stages that make up a round which is iterated 10 times for a 128-bit length key,

12 times for a 192-bit key, and 14 times for a 256-bit key. The first stage "Sub Bytes" transformation is a non-linear byte substitution for each byte of the block. The second stage "Shift Rows" transformation cyclically shifts (permutes) the bytes within the block. The third stage "Mix Columns" transformation groups 4-bytes together forming 4-term polynomials and multiplies the polynomials with a fixed polynomial mod $(x^4 + 1)$. The fourth stage "Add Round Key" transformation adds the round key with the block of data.

Details on AES can be had from [7] and [18].

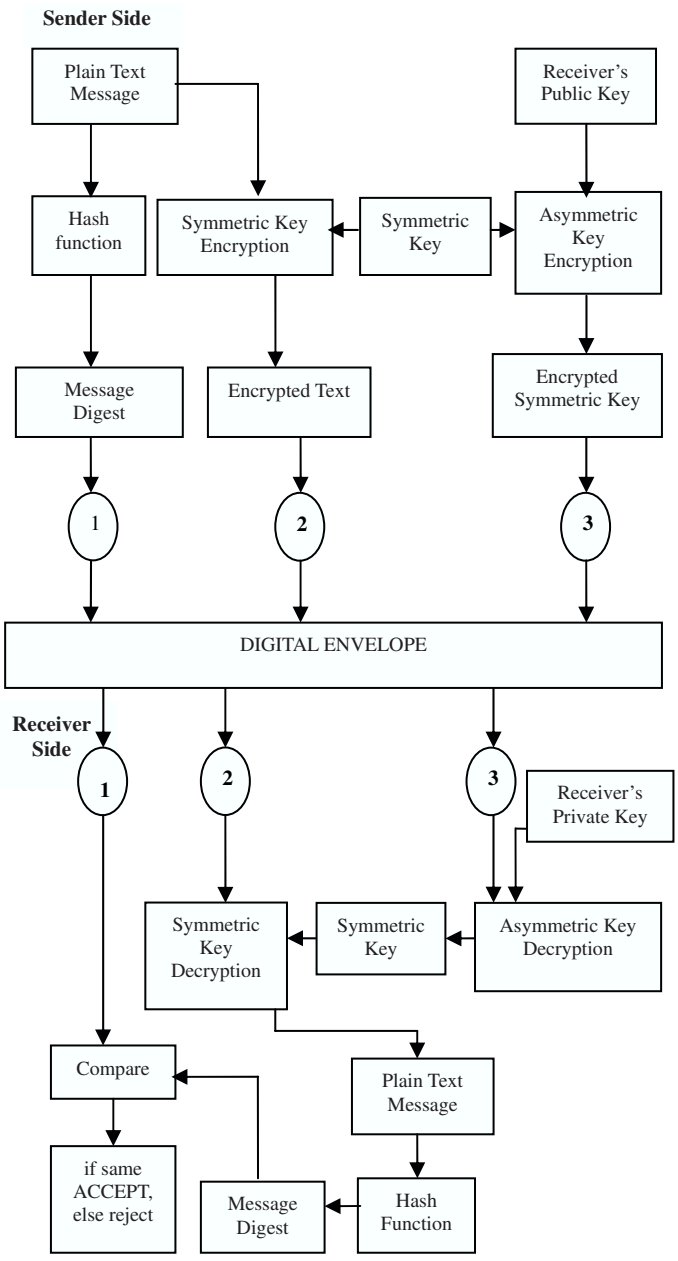

Figure 1: Digital envelope

3 Hyper-Elliptic Curve Cryptosystems

3.1 Overview of Hyper-elliptic Curve

The equation for a hyper-elliptic curve (C) is given as [13]:

$$\begin{aligned} C : y^2 + h(x)y &= f(x), h, f \in K[x], \\ deg(f) &= 2g + 1, deg(h) \leq g, \end{aligned}$$

where, f is monic and genus $(g) = (deg(f) - 1)/2$.

Unlike elliptic curves, points on hyper-elliptic curves do not form a group. Hence, a group law is defined via the Jacobian variety of C over a field K, which is a finite abelian group.

Thus, a Hyper-Elliptic Curve (HEC) over Finite Field F_p is defined as:

$$\begin{aligned} C : y^2 + h(x)y &= f(x)(\mathrm{mod} p), h, f \in K[x], \\ deg(f) &= 2g + 1, deg(h) \leq g. \end{aligned}$$

3.2 Jacobian of Hyper Elliptic Curve

The Jacobian of the curve C is the quotient group $J = D^o/P$, where D is the set of divisors of degree zero, and P is the set of divisors of rational functions. The equivalence classes of the Jacobian are represented by a unique reduced divisor (which is represented using Mumford representation) upon which we perform the group law.

3.3 Mumford Representation

Let g be the genus of a hyper elliptic curve $C : y^2 + h(x)y = f(x)$. Each nontrivial divisor class over the field K can be represented via Mumford representation $(u(x), v(x))$, where $u(x)$ and $v(x)$, $u, v \in K[x]$, are unique pair of polynomials satisfying the constraints of

- u is monic;
- deg $v <$ deg $u \leq g$;
- $u \mid v^2 + vh$ - f.

Various mathematical operations can be carried out on these hyper-elliptic curves. Details can be had from [4, 5, 12, 13, 14, 19].

3.4 Algorithm for a Hyper-elliptic Curve Cryptosystem

The basis for the Hyper-elliptic curve cryptosystem (HECC) is the Discrete Logarithm Problem which is described as follows:

"Let F_q be a finite field with q elements. Given 2 divisors, D_1 and D_2 in the Jacobian, determine $m \in Z$, such that $D_2 = mD_1$."

The following section describes the proposed HECC algorithm which exploits ElGamal technique for key generation process, encryption and decryption process which is named as HEC-ElG Algorithm (HEC-ElGA).

3.5 Algorithm for Public Key & Private Key Generation

Input: The public parameters are hyper-elliptic curve C, prime p and divisor D.

Output: The Public key P_A and Private key a_A.

Process:

1) $a_A \in_R N$ [choose a prime (a_A) at random in N];

2) $P_A \leftarrow [a_A]\ D$;
[The P_A is represented using Mumford representation which is of the form $(u(x), v(x))$];

3) return P_A and a_A.

For the random prime number generation in Step 1, one can apply the probabilistic test of Rabin-Miller [15] or the deterministic test of AKS [11]. However, various researches have proved that it takes exponential time to determine the given large number is prime or not using AKS algorithm.

3.6 Encryption/Decryption Algorithm

In this section, we present the methodology for encryption and decryption. The message 'm' that is to be sent will be encoded as a series of points represented as $(u(x), v(x))$. The encoded message is referred as E_m. For the encryption and decryption process using HECC, we have used ElGamal method to design HEC-ElG Algorithm (HEC-ElGA). Details on ElGamal method can be had from [2]. The algorithm works as follows: To encrypt and send a message to B, A performs the following steps.

- $k \in_R N$ (choose k as a random positive prime number in N);
- $Q \leftarrow [k]D$ (D is the Divisor of the HEC & The form of Q is $(u(x), v(x))$);
- $P_k \leftarrow [k]P_B$ ($P_B : (u(x), v(x))$ is receiver's($B's$) public key);
- $C_m \leftarrow \{Q, E_m + P_k\}$ (C_m: $(u(x), v(x))$ is the Cipher Text to be sent).

To decrypt the Cipher Text C_m, B extracts the first coordinate 'Q' from the cipher text then multiply with its Private Key (a_B) and subtract the result from the second coordinate. This can be written as follows:

$$\begin{aligned} E_m + kP_B - a_B(Q) &= E \\ &= E_m + kP_B - k(a_B D) \\ &= m + kP_B - a_B(kD) \\ &= E_m + kP_B - kP_B \\ &= E_m. \end{aligned}$$

In the above process, 'A' has masked the message Em by adding kP_B to it. Nobody but 'A' know the value of k, so even though P_B is a public key, nobody can remove the mask kPB. For an attacker to remove message, the attacker would have to compute k from the given D and [k]D, i.e. Q, which is assumed very hard.

Avanzi M [1] has proved that HECC over prime field is satisfactory enough to be considered as a valid alternative to elliptic curves, especially when large point groups are desired. Fan and Gong [6] also proved that HECC provides greater efficiency than either integer factorization systems or discrete logarithm systems, in terms of computational overheads, key sizes, and bandwidth. In this work, we have adopted hyper-elliptic curve for genus 2 over $GF(p)$ and have implemented the system.

4 Hashing and Message Digest

A hash function takes a message of any length as input and produces a fixed length string as output, sometimes termed a message digest [9].

The characteristics of a good hash function are:

- Should avoid collisions.
- Should try to spread keys evenly in the array.
- Should be easy to compute.

The two most-commonly used hash functions are MD5 and SHA-1.

MD5 (Message-Digest algorithm 5), is an Internet standard [17] and is one of the widely used cryptographic hash function with a 128-bit message digest. This has been employed in a wide variety of security applications.

The main MD5 algorithm operates on a 128-bit, divided into four 32-bit words. These are initialized to certain fixed constants. The main algorithm then operates on each 512-bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed rounds; each round is composed of 16 similar operations based on a non-linear function, modular addition, and left rotation.

5 Digital Envelope Combining AES & HECC

The algorithm we present here combines the best features of both symmetric and asymmetric encryption techniques. The data (plain text) that is to be transmitted is encrypted using the AES algorithm. The AES key which is used to encrypt the data is encrypted using HECC.

To ensure integrity of the data that is transmitted, the data is subjected to MD5 hash algorithm. The message digest obtained by this process is also encrypted using HECC technique. Thus the sender sends:

1) Cipher text of the message;
2) Ciphertext of the AES key, and

3) Ciphertext of the message digest.

The receiver upon receiving:

1) Cipher text of the message;

2) Ciphertext of the AES key, and

3) Ciphertext of the message digest, first decrypts the Ciphertext of the AES key to obtain the AES key.

This is then used to decrypt the cipher text of the message to obtain the plain text. The plaintext is again subjected to MD5 hash algorithm. This process yields a message digest. The ciphertext of the message digest is decrypted using HECC technique to obtain the message digest sent by the sender. This value is compared with the computed message digest. If both of them are equal, the message is accepted else rejected. Figure 2 shows the sequence of operation of the digital envelope using AES and HECC.

The entire implementation was carried out in Java 2, Standard Edition (J2SE) v 1.4.0. J2SE has the built-in classes for AES, and MD5. $generatekey(\cdot)$ is the method of KeyGenerator class which is used to generate secret key for the AES. Cipher is the class which is initialized either as encrypt mode or decrypt mode to perform relevant operation. $doFinal(\cdot)$ method of Cipher class performs the actual encryption or decryption process based on the mode initialized. The same method is used to generate the message digest. To use these classes, one has to import java.security, javax.crypto, javax.crypto.spec packages. Details on java security architecture can be had from [16].

We developed methods in Java for hyper-elliptic curve generation, base point generation, eys (both public and private) generation and encryption and decryption using HEC-ElG Algorithm (HEC-ElGA). HyperellipticCurve is the class which is used to generate hyper-elliptic curve. The divisor is generated by the Div class. The Cantor's algorithm [3] is implemented in this class to perform the group addition.

The key generation, encryption and decryption algorithm which are stated in the Section 4.5 is also implemented. We used the Java class of BigInteger to handle large integers and the method of IsProbablePrime to determine whether the large integer is prime or not. The software was run on Celeron machine @ 1.0 GHz and 256 MB RAM.

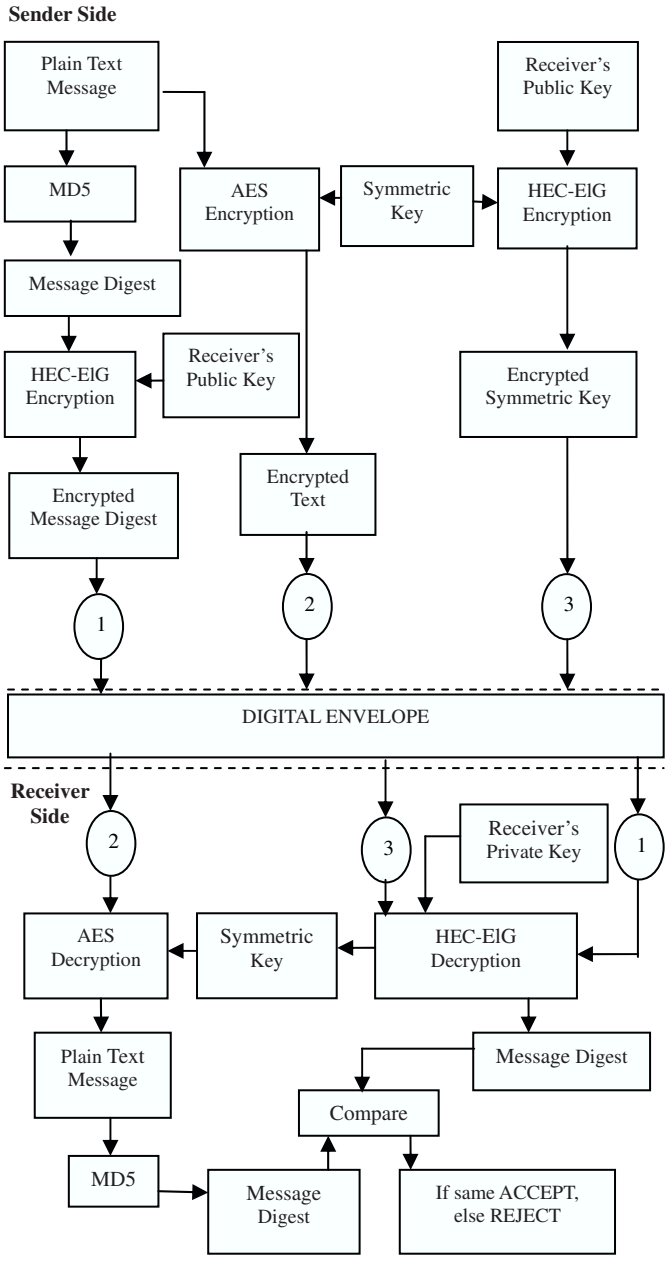

Figure 2: Digital envelope using AES & HECC

6 Results

This hybrid algorithm is tested with file data of sizes 50kb, 100kb and 150kb respectively. The key sizes considered here are recommended by the NIST and is shown in Table 3. Table 1 provides details on the time taken for encryption, decryption for AES and Triple-DES (T-DES) and also computes message digest using MD5. Table 2 provides information on Encryption & Decryption of 128

bit AES key and MD5 message digest using HECC of genus 2 over prime field (80 bits) and RSA (1024 bits). Figures 3 & 4 gives the graphical output of the results.

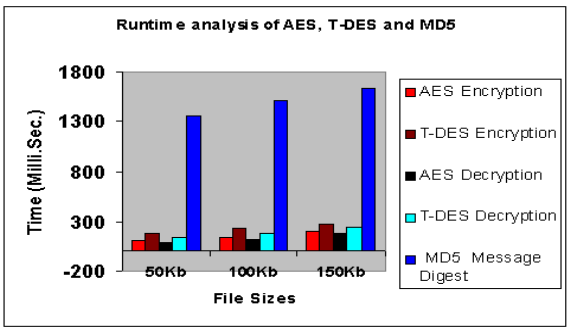

Figure 3: Runtime analysis of AES, T-DES, and MD5

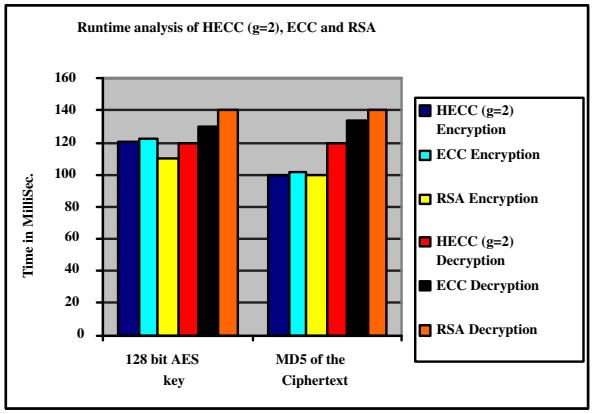

Figure 4: Runtime analysis of HECC, ECC, and RSA

7 Analysis

With any cryptographic system dealing with 128 bit key, the total number of combination is 2^{128}. The time required to check all possible combinations at the rate of rate 50 billion keys / second is approximately 5 x 10^{21} years. Moreover, the NIST recommended key sizes for various encryption techniques are given in Table 3.

From Table 3 it is clear that HECC fairs better than ECC and RSA in terms of security level. The Table 1 shows that the performance of AES is better than T-DES. Also, from Table 2, it is clear that HECC is superior to ECC and RSA. So, the digital envelope combining AES and HECC is the better alternative security mechanism for the secure e-commerce channel to achieve privacy, authentication, integrity and non-repudiation.

Table 3: NIST recommended key sizes

Symmetric Key Size (bits)	RSA Key Size (bits)	Elliptic Curve Key Size (bits)	Hyper-elliptic Curve (g=2) Key Size (bits)
80	1024	160	80
112	2048	224	***
128	3072	256	***
192	7680	384	***
256	15360	521	***

8 Conclusion

In this paper, we have designed and implemented a digital envelope in Java combining the best of both symmetric (AES) and asymmetric (HECC over GF(p) of Genus 2) methodologies. We have tested the algorithm for various sizes of files. To ensure integrity of the data, we have adopted the MD5 hash algorithm.

In this work, we have designed and implemented HEC-ElG Algorithm (HEC-ElGA) for the key generation, encryption and decryption processes and also, adopted a probabilistic primality checking ($isProbablePrime(\cdot)$ in Java) to determine whether the given number is prime or not. To determine whether the given number is prime or not, one has to use AKS algorithm which is deterministic in nature. The main limiting factor in AKS algorithm is the overhead it puts on computation in terms of time and processor speed. To illustrate, for a 7 digit number, researches have indicated that AKS algorithm takes about 20 hours whereas Rabin-Miller gives its verdict in less than a second. Finally, NIST recommended 80-bits key size for hyper-elliptic curve whereas 1024-bits and 160-bits for RSA and ECC respectively.

Therefore, one can use HECC asymmetric key technique as an alternative to ECC and RSA in the digital envelope.

References

[1] M. Avanzi, "Aspects of hyper-elliptic curves over large prime fields in software implementations," *Cryptographic Hardware and Embedded Systems*, vol. 3156, pp. 148-162, 2004.

[2] R. M. Avanzi, and L. Tanja, *Introduction to Public key cryptography from Handbook of Elliptic and Hyper elliptic curve cryptography eds. Henri Cohen*, Gerhard Frey, Chapman and Hall/CRC, Taylor and Francis, Florida, 2006.

[3] B. S. K. Jr., C. K. Koc, and C. Paar, "Cryptographic hardware and embedded systems," *4th International Workshop, Bedwood Shores*, pp. 400-414, CA, USA, 2002.

[4] S. Duquesne, and T. Lange, *Arithmetic of Hyper elliptic curves from Handbook of Elliptic and Hyper Elliptic Curve Cryptography* by Henri Cohen, Gerhard

Table 1: Time in milliseconds for AES, T-DES encryption and decryption and calculation of MD5 message digest

File Size	AES Encryption	T-DES Encryption	AES Decryption	T-DES Decryption	MD5 Message Digest
50Kb	103	180	90	140	1364
100kb	140	233	120	180	1505
150kb	210	280	182	240	1632

Table 2: Encryption & Decryption of 128 bit AES key and MD5 message digest using HECC, ECC and RSA

	HECC Encryption	ECC Encryption	RSA Encryption	HECC Decryption	ECC Decryption	RSA Decryption
128 bit AES key	121	123	110	120	130	140
Ciphertext of the MD5	100	102	100	120	134	140

Frey, Chapman and Hall/CRC, Taylor and Francis Group, Florida, 2006.

[5] C. O. Eigeartaigh, *A Comparison of Point Counting Methods for Hyper Elliptic Curve over Prime Fields and Field of Characteristics of 2*, Technical report, School of Computing, Dublin City University, Dublin, Ireland.

[6] X. Fan, and G. Gong, *Efficient Explicit Formulae for Genus 2 Hyperelliptic Curves over Prime Fields and Their Implementations*, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario N2L 3G1, CANADA, 2007.

[7] (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)

[8] M. S. Hwang, and C. Y. Liu, "Authenticated encryption schemes: Current status and key issues," *International Journal of Network Security*, vol. 1, no. 2, pp. 61-73, 2005.

[9] M. Ilya, *Hash functions: Theory, Attacks, and Applications*,' Microsoft Research, Silicon Valley Campus, Nov. 14, 2005.

[10] I. K. Salah, A. Darwish, and S. Oqeili, "Mathematical attacks on RSA cryptosystem," *Journal of Computer Science*, pp. 656-671, Aug. 2006.

[11] T. Jin, *Researching and Implementing on AKS Algorithm*, University of Bath, May 2005.

[12] T. Lange, "Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae," *Cryptology ePrint Archive*, Report 2002/121, 2002.

[13] A. J. Menezes, Y. H. Wu, R. J. Zuccherato, *An Elementary Introduction to Hyper Elliptic Curves*, Technical Report CORR 96-19, University of Waterloo, Ontario, Canada, Nov. 1996.

[14] Y. Sakai, K. Sakurai, "On the practical performance of hyper-elliptic curve cryptosystems in software implementation," *IEICE Transactions on Fundamentals*, vol. E83-A, no. 4, Apr. 2000.

[15] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 2nd Edition, Pearson Education, 2002.

[16] Sun Java. (http://java.sun.com /j2se/1.3/docs /guide/security/ CryptoSpec.html)

[17] RFC1321. (http:// www.scit.wlv.ac.uk/rfc /rfc13xx/ RFC1321.html)

[18] U. S. Department of Commerce / NIST, FIPS PUB 197, Specification for the Advanced Encryption Standard (AES), Nov. 2001. (http://csrc.nist.gov/encryption/aes)

[19] A. Weng, "Constructing hyper-elliptic curves of genus 2 suitable for cryptography," *Mathematics of Computation*, pp. 435-458, 2003.

R. Ganesan is a Senior Lecturer, Department of Computer Science and Applications in PSG College of Arts & Science, Coimbatore, India. He teaches courses for BSc Computer Science, BCA and Master of Computer Applications (MCA). At present he is pursuing his PhD programme in Computer Science. His research areas of interest include Elliptic and Hyper-elliptic curve Cryptography, Information System Security and Network security. He has published 6 National/International Journals. He has presented 5 articles in National /International conferences. He is an active member of IACSIT, and IAENG. He is acting as a reviewer in various International Journals in the field of computer science. University of Bradford, UK.

M. Gobi is a Senior Lecturer, Department of Computer Science and Applications in PSG College of Arts & Science, Coimbatore, India. He teaches courses for BSc Computer Science, BCA and Master of Computer Applications (MCA). At present he is pursuing his PhD programme in Computer Science. His research areas of interest include Cryptography, Java, Software Engineering and Information Systems Security.

K. Vivekanandan obtained his Ph.D in Computer Science from Bharathiar University during 1996. He has completed his post graduate in Applied Mathematics in 1981. He has worked as Computer Programmer/ System Analyst in the industry till 1986. He is with BSMED, Bharathiar University since 1986 as Associate Professor. He worked as the Faculty Member, Government College of Technology, IBRA, OMAN during the year 2008 - 2009. He served as Associate Professor, National University of RWANDA during February 1999 - July 2001. He is also a visiting faculty, National University of RWANDA. His research areas include Computer Simulation, Information Systems, Data Mining and E-Commerce. He has published 9 articles in National/International Journals. He has presented more than 55 research papers in National/International conferences. He has guided 5 PhD's and currently guiding 6 PhD's. He also guided 11 MPhil's.