International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 12, Issue 10 (October 2016), PP.22-28

An Hyper-Elliptic Curve Cryptosystem Scheme Using 3bc Algorithm

Md. Hussain Ansari[1], Deepak Kumar[2], Khustar Ansari[3]
[1,2,3]*Assistant Professor, Department Of Computer Science And Engineering, Guru Gobind Singh Educational Society's Technical Campus, Bokaro Steel City, India*

Abstract: This paper studies a public key generation in an enhanced ECC (Elliptic Curve Cryptosystem) using FPGA's at the hardware implementation. To improve the strength of encryption and the speed of processing, the public key and the private key of EC(Elliptic curve) over GF(2n) are used to form a shared private key (X,Y). And then the X is used with an initial point on HEC(Hyper-Elliptic Curve) over GF(2n) to generate session keys, which are used with 3BC (Block Byte Bit Cipher) (see [1], [6], [13]) algorithm for the data encryption. We are investigating a novel approach of software/ hardware code sign implemented in Verilog Hardware Description Language (VHDL), which produces hardware algorithm components to place onto the FPGAs, thereby creating adaptive software overlays differentiated by use of a Universal Unique Identifier (UUID) as a functional operand to a custom Arithmetic Logic Unit (ALU).

Keywords: ECC, HEC, 3BC algorithm, UUID, VHDL, FPGA.

I. INTRODUCTION

1.1. Hyper-Elliptic Curve Cryptosystem:

A hyper-elliptic curve of genus g over a field K is given by the equation

$$C: y^2 + h(x)y = f(x) \in K[x, y]$$

Where, $h(x) \in K[x]$

Is a polynomial of degree not larger than g and f(x) ∈ K[x] is a monic polynomial of degree 2g + 1 [7]. From this definition it follows that elliptic curves are hyper-elliptic curves of genus 1. In hyper-elliptic curve cryptography K is often a finite field. The Jacobian of C, denoted J(C), is a quotient group, thus the elements of the Jacobian are not points, they are equivalence classes of divisors of degree 0 under the relation of linear equivalence. This agrees with the elliptic curve case, because it can be shown that the Jacobian of an elliptic curve is isomorphic with the group of points on the elliptic curve. The use of hyper-elliptic curves in cryptography came about in 1989 from Neal Koblitz [14]. Although introduced only 3 years after ECC, not many cryptosystems implement hyper-elliptic curves because the implementation of the arithmetic isn't as efficient as with cryptosystems based on elliptic curves or factoring (RSA). The efficiency of implementing the arithmetic depends on the underlying finite field K, in practice it turns out that finite field of characteristic 2 are good choices for hardware implementations while software is usually faster in odd characteristic.

The Jacobian on a hyper-elliptic curve is an Abelian group and as such it can serve as group for the discrete logarithm problem (DLP) [30]. In short, suppose we have an Abelian group G and g an element of G, the DLP on G entails finding the integer a given two elements of G, namely g and g^a. The first type of group used was the multiplicative group of a finite field, later also Jacobians of (hyper) elliptic curves were used. If the hyper-elliptic curve is chosen with care, then Pollard's rho method is the most efficient way to solve DLP. This means that, if the Jacobian has n elements, that the running time is exponential in log (n). This makes is possible to use Jacobians of a fairly small order, thus making the system more efficient. But if the hyper-elliptic curve is chosen poorly, the DLP will become quite easy to solve. In this case there are known attacks which are more efficient than generic discrete logarithm solvers or even sub-exponential. Hence these hyper-elliptic curves must be avoided. Considering various attacks on DLP, it is possible to list the features of hyper-elliptic curves that should be avoided.

In 1988, Miller, N. Koblitz first forward hyper-elliptic curves cryptosystem as the expansion of ECC, the security of HECC is based on the discrete logarithm problem of hyper-elliptic curves on finite field, which is HECDLP(Hyper-elliptic Curve Discrete Logarithm Problem). Hyper-Elliptic Curve Cryptosystem has recently attracted some researcher's interests because it gives that appear to offer equal security for a smaller key size. HECC is a typically fast public key cryptosystem and it has much superiority and application efficiency [10-13]. There are some superiorities of Hyper-Elliptic Curve cryptosystem such as high efficiency, short key length as compared with other public key cryptosystems are as follows:

(1) Cryptosystem based on hyper-elliptic curves Jacobian group has the same security level as cryptosystem based on elliptic curves rational point group with the same group order. For the security of hyper-elliptic curves cryptosystem is based on HECDLP (Hyper-elliptic Curve Discrete Logarithm Problem).
(2) Hyper-elliptic curves cryptosystem can acquire the same security level with shorter operating parameters. For hyper-elliptic curves cryptosystem with genus of 3, if the basic finite field is 60 bits, the security level HECC is equivalent to that of ECC with 180 bits, and it is far more secure than RSA with 1024 bits (15-17).
(3) At present, the attack algorithms against hyper-elliptic curves cryptosystem with Low genus g (g<4) all prove to be in applicable with exponent complexity, as to hyper-elliptic curves cryptosystem with lower genus than 4, no effective attacking algorithms with sub-exponent complexity have been found, so the security of hyper-elliptic curves cryptosystem proves to be reliable.
(4) In HECC, a secure Jacobian group with large prime number order can be constructed on a relatively smaller basic field [18].

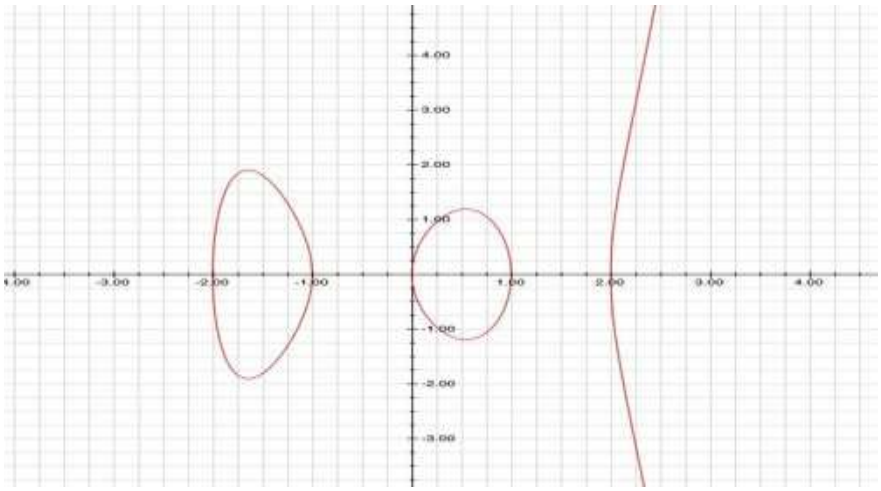


Figure 1: Hyper-elliptic Curve

1.2. Elliptic Curve Cryptography:

ECC was proposed independently by Miller [8] and Koblitz [9] in 1985, is becoming widely known and accepted. Elliptic curves are mathematical constructions, that can be defined over and field. A field is defined by a set of elements and some operations that have some special properties.

Order E is the finite field Fp on the elliptic curve, P is a point on the elliptic curve. E is defined by formula.

$$Y^2=x^3+ax+b$$

Where a, b ∈ Fp, and satisfies formula.

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

Set the order of P is a prime n, so that assemblage P is a cyclic subgroups of elliptic curves which generated by P. Prime P, elliptic curve equation E and order n constitute a public set of parameters.

The ECC key pair generation algorithm is generated by following steps:
(1) Choose a random integer d in [1,n-1].
(2) Calculate Q=d*p.
(3) Get the public key pair(Q,d).

Where d is signified private key and Q is signified public key.
To achieve the elliptic curve encryption, following steps need to do.
Expressed plaintext m as an Elliptic Curve point M.
(1) Choose a random key k in [1, n-1].
(2) Calculate C1=k*p.
(3) Calculate C2=M+k*Q.
(4) Get the public key pair (C1, C2).

Where C1 and C2 are cipher texts.
The decryption process is receiver calculate M by formula M = C2 + d * C1, then get plaintext m.

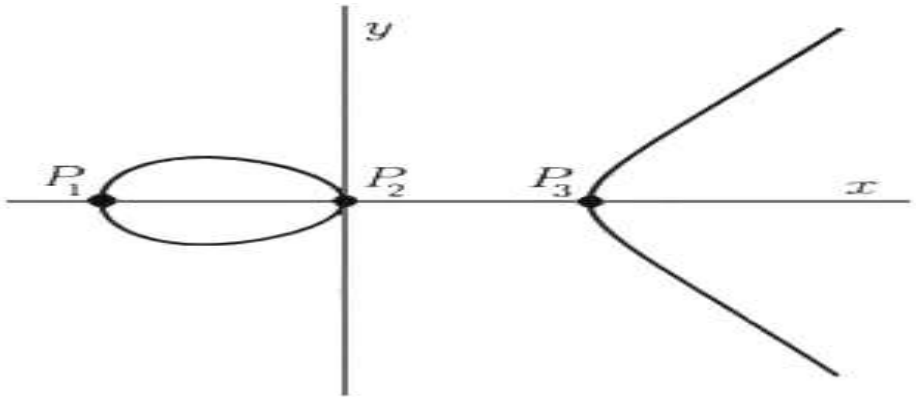


Figure 2: Elliptic curve

II. THE MATHEMATICAL OVERVIEW

2.1. Hyper-Elliptic Curve Cryptography

In the study of cryptosystem, hyper-elliptic curve has recently attracted some researchers' interests because it gives that appears to offer equal security for a far smaller key size, thereby saving the processing overhead because it gives the same security level with a smaller key length as compared to cryptosystems using elliptic curves. From the fact it is expected to be possible to use hyper-elliptic curves to factor integers, since elliptic curve method exploits the property of the Abelian groups in the same way as the cryptosystems.

A hyper-elliptic curve H of genus g(g=>1) over a field F is a nonsingular curve that is given by an equation of the following form:

$$H : v^2 + h(u)v = f(u) \text{ (in } F[u, v])$$

Where h(u) ∈ F[u] is a polynomial of degree< = g, and f(u) ∈ F[u] is a monic polynomial of degree 2g + 1.

2.1.1. Divisors:

Divisors of a hyper-elliptic curve are pairs denoted div (a(u), b(u)), where a(u) and b(u) are polynomials in $GF(2^n)[u]$ that satisfy the congruence

$$b(u)^2 + h(u)\, b(u) \equiv f(u) (\text{moda}(u))$$

They can also be defined as the formal sum of a finite number of pointson the hyper-elliptic curve. Since these polynomials could have arbitrarily large degree and still satisfy the equation, the notion of a reduced divisor is needed. In a reduced divisor, the degree of a(u) is no greater than g, and the degree of b(u) is less than the degree of a(u).

2.1.2. Reduced divisors:

Let H be a hyper-elliptic curve of genus g over a field F. A reduced divisor(definedover F) of H is defined as a form div (a, b), wherea, b ∈ F[u] are polynomial such that:
(1) a is monic, and deg b <deg a $\leq$ g,
(2) a divides $(b^2+bh - f)$.
In particular div (1, 0) is called zero divisor.

Algorithm 1: Reduction of a divisor to a Reduced Divisor.

Input: A semi-reduced divisor, D = div (a, b).

Output: The equivalent reduced divisor, D'= div D(a', b') ~ D.
1. Set a'= $(f - bh - b^2)/a$ and b'= (-h-b)(moda').

2. If $\deg_u a' > g$ then set a = a', b = b'and go to step 1.

3. Let c be the leading coefficient of a'. Set $a'=c^{-1} a'$.

4. Output D'= div (a', b').

2.1.3. Adding divisors:

If D_1= div (a_1, b_1) and D_2= div (a_2, b_2) are two reduced divisors defined over F, then Algorithm 2 finds a semi-reduced divisor or reduced divisor D_3= div (a, b).To find the unique divisor, D_3= div (a, b), Algorithm 1 should be used just after the addition of two divisors.

Algorithm 2: Addition defined over the group of divisors.

Input: Two reduced divisors, D_1 = div (a_1, b_1) and D_2 = div(a_2, b_2).
Output: A reduced divisor or semi-reduction divisor, D_3= div (a, b)
1. Compute d1, e_1 and e_2 which satisfy
d_1=GCD (a_1, a_2) and d_1= e_1a_1+ e_2a_2.

2. If d= 1, then
a:= a_1a_2, b := ($e_1a_1b_2$+$e_2a_2b_1$)mod a,

Otherwise do the following:
(1) Compute d,c_1 and $s_3$which satisfy
d = GCD (d_1, b_1 + b_2 + h) and d = c_1d_1+ s_3(b_1+ b_2+ h).
(2) Let and s_1:= $c_1e_1$and s_2:= c_1e_2, so that d= s_1a_1 +s_2a_2 +s_3(b_1+b_2+h)
(3) Leta := a_1a_2/d_2,b := ($s_1a_1b_2$+ $s_2a_2b_1$+ s_3(b_1b_2+ f))/d (mod a).
3. Output D3= div (a, b).

2.1.4. Message Encryption And Decryption

Let us suppose a text file has to be encrypted, a user can encrypt the ASCII code of each and every printable character on the keyboard, let us say he has to encrypt an 8bitnumber, canrepresent128charactersonthekeyboard.Figure shows the sequence of steps to be followed when a message to be encrypted and decrypted using Hyper-elliptic Curve Cryptography.

III. PROPOSED WORK

3.1. Encryption And Decryption With 3bc Algorithm

With 3BC algorithm, the procedure of data encryption is divided into three parts, inputting plaintext into data block, byte-exchange between blocks, and bit-wise XOR operations between data and session key.

3.1.1. Session Key Generation

As we know that the value which is obtained by multiplying one's private key by the other's public key is the same as what is computed by multiplying one's public key to the other's private key. The proposed key generation HEC with 3BC algorithm to generate session keys and cipher text. The encryption and decryption are processes. The result of generates a session key for 3BC. With this advantage and the homogeneity of the result of operations, the proposed 3BC algorithm uses a 64-bit session key to perform the encryption and decryption. Given the sender's private key Ks and the receiver's public key Pr, we multiply Pr by Ks to obtain a point KsPr =(X, Y) on HEC, where X = X_1X_2…..X_m and Y = Y_1Y_2…..Y_n. Then we form a key N by concatenating X and Y (i.e.N = X_1X_2. . .$X_mY_1Y_2$. . .Y_n), and generate the session keys as follows:

- **i.** If the length (number of digits) of X or Y exceed four, then the extra digits on the left are truncated. And if the length of X or Y less than four, then they are p added with 0's on the right. This creates a number N'= $X'_1X'_2X'_3X'_4Y'_1Y'_2Y'_3Y'_4$. Then anew number N" is generated by taking the modulus of each digit in N' with 8.
- **ii.** The first session key sk1 is computed by taking bit-wise OR operation on N" with the reverse string of N".
- **iii.** The second session key sk2 is generated by taking a circular right shift of sk1 by one bit. And repeat this operation to generate all the subsequent session keys needed until the encryption is completed.

3.1.2. Block Data Input

The block size is defined as 64 bytes. A block consists of 56 bytes for input data, 4 byte for the data block number, and 4 byte for the byte-exchange block number (see Figure). During the encryption, input data stream are blocked by 56 bytes. If the entire input data is less than 56 bytes, the remaining data area in the block is padded with each byte by a random character. Also, in the case where the total number of data blocks filled is odd, then additional block(s) will be added to make it even, and each of those will be filled with each byte by a

random character as well. Also, a data block number in sequence is assigned and followed by a byte-exchange block number, which is either 1 or 2.

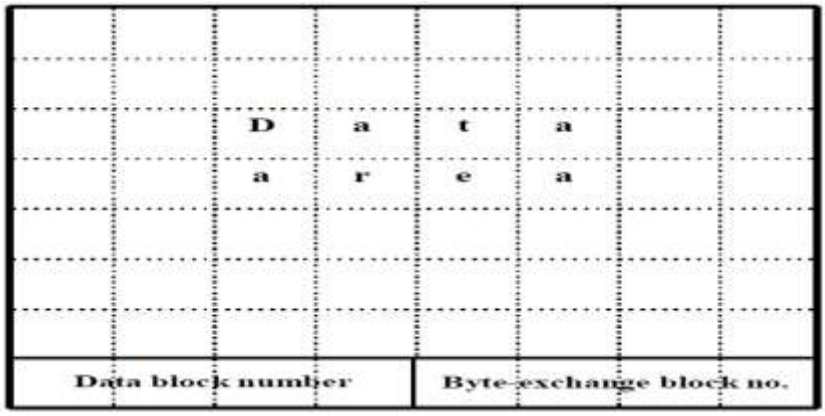


Figure: Structure of block

3.1.3. Byte-Exchanges Between Blocks

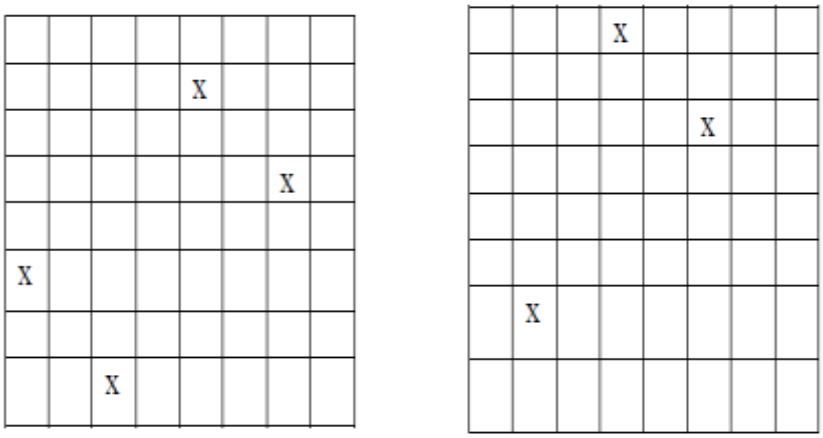
After filling the data into the blocks, we begin the encryption by staring with the first data block and select a block, which has the same byte-exchange block number for the byte exchange. In order to determine which byte in a block should be exchanged, we compute its row-column position as follows:

For the two blocks whose blocks exchange number, n = 1, we compute the following:

Byte-exchange row= (N_i* n) mod 8 (i = 1, 2 . . . 8),

Byte-exchange col = ((N_i*n) + 3) mod 8 (i = 1, 2 . . . 8),

Where N_i is a digit in N". These generate 8 byte-exchange positions. Then for n = 1, we only select the non-repeating byte position (row, col) for the byte-exchange between two blocks whose block exchange numbers are equal to 1. Similarly, we repeating the procedure for n=2. The following example illustrates the process of byte-exchange:



n = 1 **n = 2**

Figure: Exchange bytes at (row, col) for a selected pair of blocks

Example: Given the values of a sender's public key 21135 and a receiver's private key 790, we compute the position of row and col for byte-exchange as follows:

For n = 1. It follows that N"= 11357900 (after truncation, padding and concatenation), and

row = ((1, 1, 3, 5, 7, 9, 0, 0) * 1)mod 8 = (1, 1, 3, 5, 7, 1, 0, 0) and

col = (((1, 1, 3, 5, 7, 9, 0, 0) * 1 + 3)mod 8) = (4, 4, 6, 0, 2, 4, 3, 3).

This results 8 byte-exchange positions, (1,4), (1,4), (3,6), (5,0), (7,2), (1,4), (0,3) and(0,3). However, counting only once for repeating pairs, the four bytes at (1,4) (3,6), (5,0), and (7,2) will be selected for byte-exchange between two blocks (see Figure).

For n = 2, we have

row = ((1, 1, 3, 5, 7, 1, 0, 0) * 2)mod 8 = (2, 2, 6, 2, 6, 2, 0, 0) and

col = (((1, 1, 3, 5, 7, 1, 0, 0) * 2 + 3)mod 8 = (5, 5, 1, 5, 1, 5, 3, 3),

which results 8 byte-exchange positions, (2,5), (2,5), (6,1), (2,5), (6,1), (2,5), (0,3) and (0,3). Similarly, only three byte positions at (2,5), (6,1), and (0,3) are used for byte-exchanges between two blocks as shown in Figure.

3.1.4. Bit-Wise Xor Between Data And Session Keys

After the byte-exchange is done, the encryption proceeds with a bit-wise XOR operation on the first 8 byte data with the session sk1 and repeats the operation one very 8 bytes of the remaining data with the subsequent session keys until the data block is finished (see Figure).

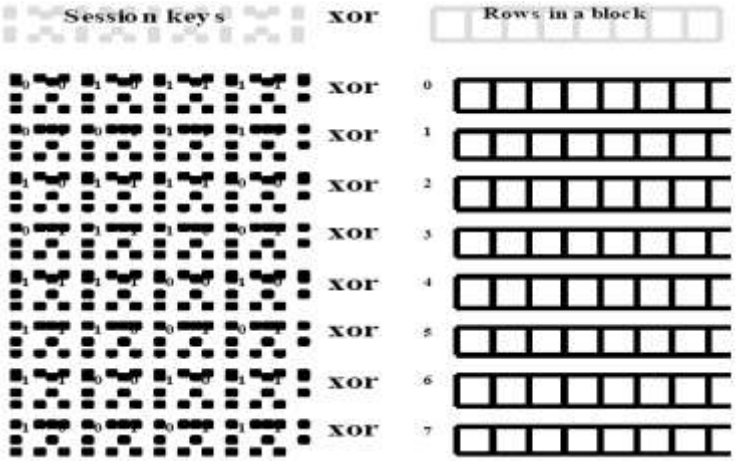


Figure: The bit-wise XOR on rows with session keys

Note that the process of byte-exchange hides the meaning of 56 byte data, and the exchange of the data block number hides the order of data block, which needs to be assembled later on. In addition, the bit-wise XOR operation transforms a character into a meaningless one, which adds another level of complexity to deter the network hackers. Figure an encryption procedure using session keys deriving from a private key and a public key.

3.1.5. Decryption:

Decryption procedures are as follows:

- **i.** A receiver generates a byte exchange block key sk1 and a bit-wise XOR key sk2 by using the sender's public key and the receiver's private key.
- **ii.** The receiver decrypts it in the input data receiving sequence. The receiver does bit-wise XOR operation bit by bit, and then, a receiver decodes cipher text by using a byte exchange block key sk1 and moves the exchange bytes back to their original position. We construct data blocks in sequence by using the decoded data block number.

IV. Conclusion

The first result is that HECC algorithm is faster than ECC algorithm and it uses smaller key size length. One of the most important problems in smart cards is limitation of storage. Providing a multipurpose smart card with smaller key size is very important. On the other hand, since in the multipurpose smart card there are three different systems, high speed is one the most important requirements in the system.

HECC algorithm has high level of security and it is faster than ECC algorithm. Since, with the addition of 3BC algorithm it is more efficient than the other cryptographic algorithm.

References

[1]. A.J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, "Handbook of Applied Cryptography", CRC Press, 1996.
[2]. W. Rankl, W. Effing, "Smart Card Applications, Design Models forusing and programming smart cards," Springer-Verlag, 2007.
[3]. A. B. Mohamed, A. A. Hamid, K. Y, Mohamed,"Implementation of an Improved Secure System Detection for E-passport by using EPCRFIDTags," 2009.
[4]. Nyamasvisva, E. Tadiwa, H. Halabi, "Multi-level security algorithm for random ZigBee Wireless Sensor Network," In: 4th International Symposium on Information Technology 2010 (ITSim'10), 15-17 June 2010, KLCC, Kuala Lumpur, 2010, pp. 612-617.
[5]. IEEE Standard Specifications for Public-Key Cryptography, Jan. 2000.
[6]. M. Savari, M. Montazerolzohour, Y. E. Thiam, "Comparison of ECC and RSA Algorithm in Multipurpose Smart Card Application", 2012.

[7]. V. S. Miller, "Use of elliptic curves in cryptography", In Advances in cryptology—CRYPTO '85, volume 218 of Lecture Notes in Computer Sci., pages 417–426. Springer, Berlin, 1986.

[8]. N. Koblitz,"Elliptic curve cryptosystems", Math. Computer, 48(177):203–209, 1987.

[9]. R. M. Avanzi, "Aspects of Hyper-elliptic Curves over Large Prime Fields in Software Implementations [A]", International Association for Cryptology Research 2004[C], Berlin, Heidelberg, New York: Springer-Verlag, 2004, 148~162.

[10]. B. Pfitzmann, M. Waidner,"Anonymous fingerprinting [A]", Advances in Cryptology-EUROCRYPT'97[C], Berlin, Heidelberg, New York: Springer-Verlag, 1997, 88~102.

[11]. M. Abe, M. Ohkubo and K. Suzuki, "1 out of n Signature from a Variety of Keys [A]", Advances in Cryptology-ASIACRYPT2002[C], Berlin, Heidelberg, New York: Springer-Verlag, 2002, 415~423.

[12]. X. Zhou, X. Y. Yang, "An Anonymous Digital Fingerprinting Scheme Based on Ring Signature", Computer Engineering, 2007, 33(21): 152-154.

[13]. N. Koblitz, "Hyper-elliptic cryptosystems" Journal of Cryptology Volume 1, Number 3, pages 139-150 1989.

[14]. X. Zhou "Improved Ring Signature Scheme Based on Hyper-Elliptic Curves" IEEE International Conference on Future Information Technology and Management Engineering, FITME pp. 373-376 2009.

[15]. X. Zhou, X. Yang and P. Wei "Hyper-elliptic curves based group signature" Control and Decision Conference, CCDC '09, Chinese pp. 2280-2284, 2009.

[16]. X. Zhou and X. Yang, "Hyper-Elliptic Curves Cryptosystem Based Blind Signature", Pacific-Asia Conference on Knowledge Engineering and Software Engineering, KESE '09, 2009.

[17]. William Stallings, "Cryptography and Network Security", Principles and Practice. ed., Prentice Hall, New Jersey, 2003.

[18]. R. Schoof, "Elliptic Curves over Finite Fields and the Computation of Square Roots mod p", Mathematics of Computation, Vol. 44, No. 170, pp. 483-494, April 1985.

[19]. F. Morain,"Building cyclic elliptic curves modulo large primes", Advances in Cryptology – EUROCRYPT 91, Lecture Notes in Computer Science, 547: 328-336, 1991.

[20]. N. Koblitz,"A Course in Number Theory and Cryptography", Springer-Verlag, second edition, 1994.

[21]. Padma Bh, D. Chandravathi, P. P. Roja, "Encoding And Decoding of a Message in the Implementation of Elliptic Curve Cryptography using Koblitz's Method", IJCSE, Vol. 02, No. 05, 1904-1907, 2010.

[22]. T. C. Lee, B. K. Lee, "A HESSL (Highly Enhanced Security Socket Layer) protocol", In: The Proceedings of the Seventh IEEE International Conference on E-Commerce Technology, July 19-22, 2005, Munich, Germany, 456-460.

[23]. B. Schneier, Description of a new variable-length key, 64-bit block cipher (blowfish), In: Proceedings, Workshop on Fast'78 Software Encryption, New York, Springer-Verlag (1993).

[24]. B. k. Lee, T. C. Lee, S. H. Yang, "An ASEP (Advanced Secure Electronic Payment) Protocol Design Using 3BC and ECC(F_2^m) Algorithm, the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04), 0-7695-2073-1/2004.

[25]. B. k. Lee, T. C. Lee, S. H. Yang "HESSL (Highly Enhanced Security Socket Layer) Protocol",the Seventh IEEE International Conference on E-Commerce Technology (CEC'05), 1530-1354/2005.

[26]. David G. Cantor, "Computing in the Jacobian of a Hyper-elliptic Curve", Mathematics of Computation, Volume 48, Number 177, January 1987, Pages 95-101.

[27]. Tai-Chi Lee, "A Public Key Generation in an Enhanced Elliptic Curve Cryptosystem over Gf(2^n)", International Journal of Pure and AppliedMathematics, Volume 78 No. 6, 2012, page 831-848.

[28]. D. G. Cantor, "Computing in the Jacobian of a hyper-elliptic curve", Mathematicsof Computation 48 (1987), 95-101.

[29]. N. Koblitz, "Algebraic Aspects of Cryptography, Algorithms and Computation inMathematics", Vol. 3, Springer-Verlag, New York, 1998.

[30]. Mumford, D. Tata Lectures on Theta. II. Boston, MA: Birkhuser, 1984.