Software Implementation of Curve based Cryptography for Constrained Devices

Kakali Chatterjee
Delhi Technological University
(Formerly Delhi College of Engineering)
Delhi, India

Asok De
Ambedkar Institute of Technology
Delhi, India

Daya Gupta
Delhi Technological University
(Formerly Delhi College of Engineering)
Delhi, India

ABSTRACT

Curve based cryptography are preferred for embedded hardware since they require shorter operand size than RSA to attain the same security level. So ECC and HECC are more suitable in constrained environment such as smart cards if we can select suitable curves and efficient scalar multiplication technique to speed up arithmetic on the curve. With this in view, this paper explores in details the main operations like scalar multiplication, group operations on Jacobian, finite field operations etc which are the prime steps for efficient implementation of ECC / HECC. We also have compared the timings of main operations like scalar multiplication, encryption and decryption of Elliptic and Hyperelliptic curve cryptosystems to study the relative performance of these cryptosystems.

Keywords

Elliptic Curve Cryptography (ECC), Hyperelliptic Curve Cryptography (HECC), Scalar Multiplication

1. INTRODUCTION

Digital communication nowadays mainly based on Public-Key Cryptosystems because they allow secure communications over insecure channels without prior exchange of a secret key and they also enable digital signatures. Majority of the products and standards that employ public key cryptography for encryption and digital signatures use RSA. But RSA is not suitable for constrained devices because of long key length. Curve based cryptography are attractive to designers of embedded hardware since they require smaller fields than RSA to attain the same security level. This fact makes curve based cryptography namely ECC and HECC a very good choice for platforms with limited resources. The purpose of this paper is the performance study of ECC and HECC, which are based on curve arithmetic and offer significant benefits over RSA when used in constrained environment such as smart card.

Neal Koblitz [1] and Victor Miller proposed Elliptic Curves Cryptography (ECC) based on the discrete logarithm problem on elliptic curves over finite fields in mid-1980s. Subsequently Hyperelliptic Curve Cryptography (HECC) was proposed by Koblitz [2] in 1989 based on the discrete logarithm problem on the Jacobian of hyperelliptic curves over finite fields. For HECC (or ECC) two types of fields are being considered i.e. binary and prime fields for practical implementations on both types of software platforms like general purpose or embedded processors and hardware devices, such as FPGAs.

ECC and HECC seem to be specially promising for the use in embedded environments where memory and speed is constrained. The suitability for constrained systems results from the short operand sizes of ECC and HECC compared to other public key schemes, e.g. RSA [Rivest et al. 1978] or DL based systems. It is widely accepted that for most cryptographic applications based on EC or HEC, the necessary group order is of size at least $\approx 2^{160}$. Thus, for HECC over F_q we will need at least g. $\log_2 q \approx 2^{160}$, where g is the genus of the curve. Therefore, we will need a field order $q \approx 2^{40}$for genus 4, $q \approx 2^{54}$ for genus 3, and $q \approx 2^{80}$ for genus 2 HEC. Hence, one needs 40-bits to 80-bit long operands to compute the group operations for these curves. In the case of ECC we have to work with operand lengths of approximately 160 bits whereas in the case of RSA, the operands will be approximately 1024 bits in order to achieve the same security. ECC and HECC are therefore more suitable for implementation in the constrained platforms like the PDA, smartcard, handheld devices etc.

While implementing an ECC/ HECC system, several choices are to be made. These include selection of Protocols & Standards, choice of curves, choice of coordinates for group operations, representation of a scalar multiplication algorithm, finite field operations like addition, multiplication etc. Selection of the factors can be influenced by application platform, constraints of particular computing environment like processing speed, code size, memory size etc and communicational constraints like bandwidth, response time [3].

In this paper we have explored in details the main operations like scalar multiplication, group operations on Jacobian, finite field operations etc which are the prime steps for efficient implementation of ECC / HECC. We have implemented ECC (genus 1) and HECC (genus 2) on different binary fields and also compare the scalar multiplication, encryption and decryption timing of ECC / HECC.

The rest of the Paper is organized as follows:

In Section 2, Mathematical Background is discussed; Section 3 provides hierarchy of operations of Curve based Cryptography; Section 4 presents Implementation Results; Finally we conclude the paper in Section 5.

2. MATHEMATICAL BACKGROUND

2.1 Arithmetic of Elliptic Curve Cryptography

All elliptic curves can be written in Weierstrass form [4] as follows:

$$E: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Where the coefficients $a_1, a_2, a_3, a_4, a_6 \in K$ and $\Delta \neq 0$, where Δ the discriminant of E.

Let $\mathbb{F}_q$ denote a finite field of characteristic p, i.e. $q = p^{\delta}$ with p prime. It is possible to define the notion of an elliptic curve over any field by giving a general equation.

Case-I. If $p = 2$ then an elliptic curve E defined over $\mathbb{F}_q$ is given by an equation

$y^2 + xy = x^3 + ax^2 + b,$

where $a, b \in K$ and $b \neq 0$. For every field K containing $\mathbb{F}_q$ (so in particular for $K = \mathbb{F}_q$) one considers the set

$E(K) := \{(x, y) \in K \times K \mid y^2 + xy = x^3 + ax^2 + b\} \cup \{\infty\}$.

This curve is non supersingular curve and discriminant of E is $\Delta = b$.

Case-II. If $p > 2$ then an elliptic curve E defined over $\mathbb{F}_q$ is given by an equation $y^2 = x^3 + ax + b$, where $a, b \in K$ and $4a^3 + 27b^2 \neq 0$. For every field K containing $\mathbb{F}_q$ one now considers the set

$E(K) := \{(x, y) \in K \times K \mid y^2 = x^3 + ax + b\} \cup \{\infty\}$.

Here with this definition, we consider the elliptic curves where discriminant of E is $\Delta = -16(4a^3 + 27b^2)$. We have considered this curve equation for our implementation.

The set $E(K)$ of rational points on E defined over a field K is an abelian group, where the operation (generally denoted additively) is defined by the well-known law of chord and tangent, and the identity element is the special point ∞, called the point at infinity.

2.2 Arithmetic of Hyperelliptic Curve Cryptography

We can define hyperelliptic curves as an algebraic curve Let $\mathbb{F}$ be a finite field, and let $\overline{\mathbb{F}}$ be the algebraic closure of $\mathbb{F}$ [5]. A hyperelliptic curve C of genus $g > 1$ over $\mathbb{F}$ is the set of solutions $(u, v) \in \mathbb{F} \times \mathbb{F}$ to the equation $C: v^2 + h(u)v = f(u)$. The polynomial $h(u) \in \mathbb{F}[u]$ is of degree at most g and $f(u) \in \mathbb{F}[u]$ is a monic polynomial of degree $2g + 1$. For odd characteristic it suffices to let $h(u) = 0$ and to have f(u) square free. If no point on the curve over the algebraic closure $\overline{\mathbb{F}}$ of $\mathbb{F}$ satisfies both partial derivatives $2v + h(u) = 0$ and $h'(u)v - f'(u) = 0$, then the curve is said to be non-singular.

A divisor $D = \sum m_i P_i, m_i \in \mathbb{Z}$, is a finite formal sum of $\overline{\mathbb{F}}$ points. Its degree is the sum of the coefficients $\sum m_i$. The set of all divisors form an Abelian group denoted by $D(C)$. The set of degree zero divisors D^0 forms a subgroup of $D(C)$.

Every rational function consisting of the formal sum of the poles and zeros of the function on the curve C gives rise to a divisor of degree zero. Such divisors are called principal and the set of all principal divisors is denoted by P. If $D_1, D_2 \in D^0$ then we write $D_1 \sim D_2$ if $D_1 - D_2 \in P$; D_1 and D_2 are said to be equivalent divisors. Now, we can define the Jacobian of C as the quotient group D^0/P [6].

If we want to define the Jacobian over $\mathbb{F}$, denoted by $\mathbb{J}_C(\mathbb{F})$, we say that a divisor $D = \sum m_i P_i$ is defined over $\mathbb{F}$ if $D^{\sigma} = \sum m_i P_i^{\sigma}$ is equal to D for all automorphisms σ of $\overline{\mathbb{F}}$ over $\mathbb{F}$. Cantor shows that each element of the Jacobian can be represented in the form $D = \sum_{i=1}^{r} P_i - r.\infty$ such that for all $i \neq j$, P_i and P_j are not symmetric points. Such a divisor is called a semi-reduced divisor. Each element of the Jacobian can be represented uniquely by such a divisor, subject to $r \leq g$. Such divisors are referred to as reduced divisors. We use the reduced divisor in addition of $\mathbb{J}_C$.

3. HIERARCHY OF OPERATIONS OF CURVE BASED CRYPTOGRAPHY

The hierarchy of operations for ECC and HECC, can be divided in three levels as shown in Figure 1. The highest level shows the main operation in any curve-based primitive that is the scalar multiplication. At the next level are the point/divisor group operations in different co-ordinates. The lowest level consists of finite field operations such as addition, subtraction, multiplication and inversion required to perform the group operations. The main difference between ECC and HECC is in group operation because these consist of different sequences of operations. Unlike elliptic curves, the points on the hyperelliptic curve do not form a group. The additive group on which the cryptographic primitives are implemented is the divisor class group. Each element of this group is a reduced divisor. HECC are a bit more complex when compared with the ECC point operation, but they use shorter operands.

3.1 Scalar Multiplication of curves

Fast scalar multiplication is crucial in some environments such as in hand-held devices with low computational power. Different efficient scalar multiplication methods in elliptic curves like Windows-NAF, Fixed-Base comb, Montgomery point multiplication etc are available. Efficient techniques for High Speed scalar multiplication is found in [7]. Some efficient and innovative methodology for accelerating the elliptic curve point formulae over prime fields are proposed in [8] [9]. We proposed a secure access of smart cards using elliptic curve cryptography in [10]. In cases of HECC the main operations such as key agreement and signing/verifying involve scalar multiplication using a large integer. Any algorithm for scalar multiplication requires an efficient method of performing arithmetic in the Jacobian. This arithmetic essentially consists of two operations - addition and doubling of divisors. There has been extensive research to obtain cost effective 'explicit formula' for performing addition and doubling on a HEC over $\mathbb{F}_{2^n}$.For elliptic/ hyperelliptic curves, most useful method for scalar multiplication is using Montgomery's ladder as it can resist side channel attacks.

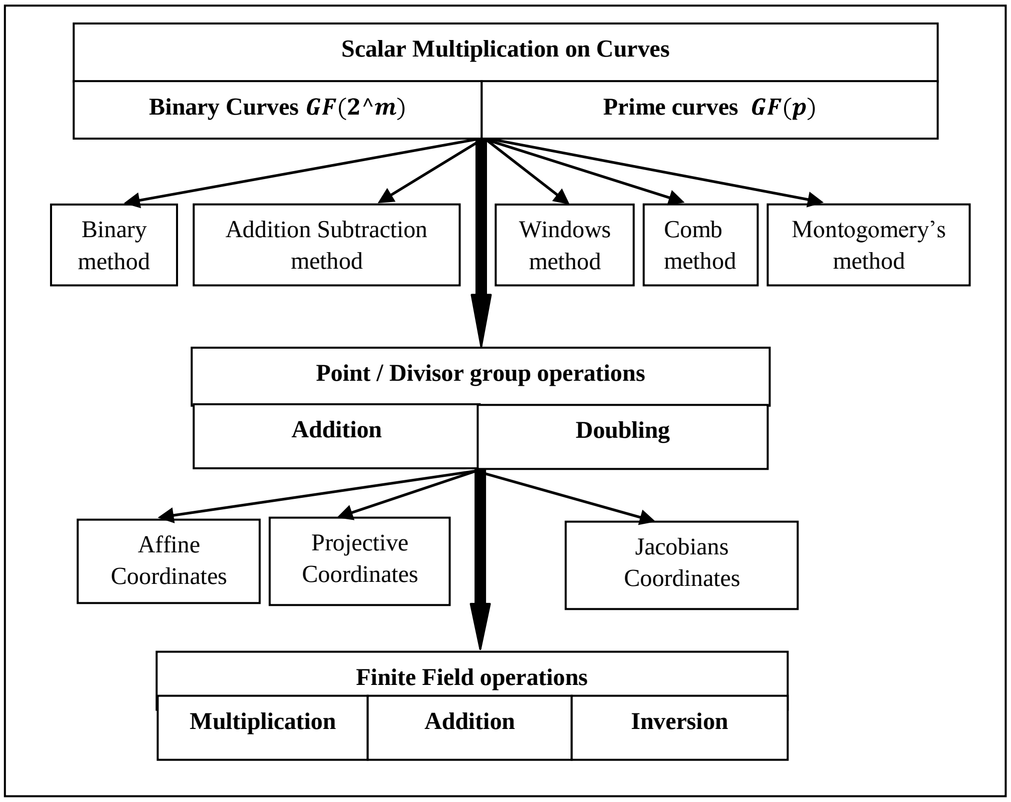

Fig 1: Hierarchy of operations of curve based cryptosystems

We have explored the scalar multiplication cost found in [11] for different co-ordinate systems which are given in Table 1. From Table 1, we can say that Affine Coordinate is preferable for general Addition and Doubling operation.

The first explicit formula for genus 2 proposed by Harley have been followed by work of Lange [12], [13]. After extensive research on explicit formula for performing addition and doubling, Avanzi [14] proposes a software implementation of genus 2 and 3 hyperelliptic curves over large prime fields. Pelzl and Wollinger [6],[15],[16] propose a cost effective explicit formula for genus 2 and 3 curves and give first implementation of a HEC cryptosystem on an embedded processor. We have discussed Evolution of Hyperelliptic Curve Cryptosystems in [17].

3.2 Group operations on a Jacobian

Cantor's algorithm [18] is used for doing arithmetic in general hyperelliptic curve which applies to any genus and characteristic. This transfer the group laws in a sequence of Composition and Reduction using only polynomial arithmetic.

Group operations on a Jacobian are performed in two steps: addition of generic divisors and doubling of generic divisors. Addition of divisor classes means multiplication of ideal classes, which consists in a composition of the ideals and a first reduction to a basis of two polynomials. The output of this algorithm is in semi-reduced form. Then the second algorithm (reduction) is used to find the unique representative in the class.

Algorithm 1 (Composition)

```
Input: $D_1 = [u_1, v_1]$, $D_2 = [u_2, v_2]$. $C: y^2 + h(x)y = f(x)$
Output: $D \sim D_1 + D_2$ where $D = [u, v]$
1. Compute $d_1 = \gcd(u_1, u_2) = e_1u_1 + e_2u_2$;
2. Compute $d = \gcd(d_1, v_1 + v_2 + h) = c_1d_1 + c_2(v_1 + v_2 + h)$;
3. Let $s_1 = c_1e_1, s_2 = c_2e_2, s_3 = c_2$;
4. $u = u_1u_2/d^2$;
   $v = \{s_1u_1v_2 + s_2u_2v_1 + s_3(v_1v_2 + f)\}/d \mod u$
```

Algorithm 2 (Reduction)

```
Input: $D = [u, v]$ semi reduced
Output: $D' = [u', v']$ reduced with $D \sim D'$
1. let $u' = (f - vh - v^2)/u$, $v' = (-h - v) \mod u'$;
2. if $\deg u' > g$ put $u = u'$, $v = v'$;
   goto step 1;
3. make $u'$ monic.
```

Harley's algorithm for genus 2 Curve: Cantor's algorithm is slow due to Polynomial arithmetic. The solution is to transform polynomial operations into field operations (explicit formula) by considering most frequent cases (occur with probability ~1- O (1/q)). It was done by Harley in 2000 [19] by using reduced divisors represented by Mumford's representation for input and output divisor classes on genus 2 curves.

Table 1: Cost of Addition and Doubling operations in Elliptic and Hyperelliptic Curves

Curves	**Elliptic Curves (defined over F_p)**		**Hyperelliptic Curves in odd characteristic**		**Hyperelliptic Curves in even characteristic**	
Coordinate System	**Addition**	**Doubling**	**Addition**	**Doubling**	**Addition**	**Doubling**
Affine (*A*)	I+2M+S	I+2M+2S	I+22M+3S	I+22M+5S	I+22M+3S	I+20M+6S
Projective (*P*)	12M+2S	7M+5S	47M+4S	38M+6S	49M+4S	38M+7S
New co-ordinates (*N*)	12M+4S (Jacobian coordinates)	4M+6S (Jacobian coordinates)	47M+7S	34M+7S	48M+4S	37M+6S

3.3 Finite Field operations on curves

Field addition, multiplication, squaring and inversion are the basis for the group operations on elliptic and hyperelliptic curves. Adding elements in F_2^n is accomplished by a bitwise XOR of the components. A field multiplication of $a, b \in \mathbb{F}_2^n$ can be accomplished by first multiplying a and b as integers and then reducing the result modulo p. In the polynomial multiplication divide and conquer method of Karatsuba- Ofman is adapted [4]. For reducing a binary polynomial obtained by multiplying two binary polynomials of degree $\leq m-1$, or by squaring a binary polynomial of degree $\leq m-1$, Barrett's method for reduction is one of the popular method for polynomial reduction. Montgomery's algorithms for multiplication & reduction is also found in literature as it is efficient in restricted devices [11]. The Extended Euclidean Algorithm is applied to calculate inverses efficiently in $\mathbb{F}_2^n$.

4. IMPLEMENTATION RESULTS

Alfred J.Menezes proposed an elementary introduction of genus 2 HECC and improved the Cantor's algorithm for adding Jacobian of hyperelliptic curve in [5]. According to his proposition, we find a semi reduced divisor $D = div(a, b)$ with $a, b \in K[u]$, such that $D \sim D_1 + D_2$ where $D_1 = div(a_1, b_1)$, $D_2 = div(a_2, b_2)$. A semi reduced divisor $D = div(a, b)$ defined over a finite field K is taken as input to find out the (unique) reduced divisor $D' = div(a', b')$ such that $D' \sim D$. We have considered the hyperelliptic curve $C: v^2 + (u^2 + u)v = u^5 + u^3 + 1$ of genus 2 over the finite field $\mathbb{F}_2^5$. The semi-reduced divisor D will be computed by computing $D_1 + D_2$.

We have examined the performance of HECC on a PC with Intel Core 2DUO CPU T6400@2.00GHz with 4GB RAM and windows vista operating system using jdk1.6. We have performed scalar multiplication technique (Montgomery's algorithms) for Hyperelliptic genus 2 curves (binary fields) in affine co-ordinates when group orders are 2^162, 2^166, 2^176, 2^182. ECC scalar multiplication (using binary method) timings are also shown in Table 2 with hyperelliptic curves timing. EC scalar multiplication have been done on NIST recommended elliptic curves over $\mathbb{F}_2^n$ where n=163, 233 and 283.

Our result shows that HECC scalar multiplication takes less time than ECC scalar multiplication and the variation is shown in Table 2.

Table 2. Timing of scalar multiplication operations in Elliptic and Hyperelliptic Curves

Curves	**Field**	**Group order**	**Scalar Multiplication (ms)**
Hyperelliptic Curves (genus 2)	$\mathbb{F}_2^{81}$	2^{162}	2.12
	$\mathbb{F}_2^{83}$	2^{166}	2.40
	$\mathbb{F}_2^{88}$	2^{176}	2.51
	$\mathbb{F}_2^{91}$	2^{182}	2.86
Elliptic Curves (genus 1)	$\mathbb{F}_2^{163}$	2^{163}	4.24
	$\mathbb{F}_2^{233}$	2^{233}	8.61
	$\mathbb{F}_2^{283}$	2^{283}	12.72

Comparison of Scalar Multiplication time of ECC and HECC are shown in Figure 2.

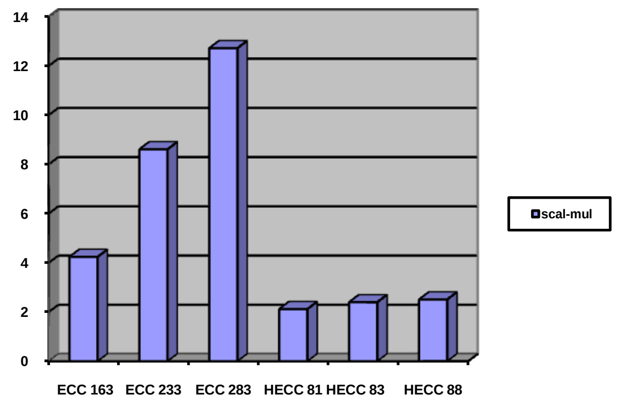

Fig-2 – Comparison of Scalar Multiplication time of ECC and HECC

Table 3: Comparison of ECC and HECC for equivalent key sizes

Operations (ms) (Binary field)	ECC (genus 1)			HECC (genus 2)			
	Field order 2^ 163	Field order 2^233	Field order 2^ 283	Field order 2^81	Field order 2^83	Field order 2^88	Field order 2^91
Encryption	797	882	928	668	893	928	965
Decryption	281	385	400	191	224	257	325

Next we have done experiment for encryption and decryption of a Text File (File size :899 Bytes) using elliptic and hyperelliptic curve cryptography to compare the timings. The results are enlisted in Table 3.

Comparison of Encryption and Decryption time of ECC and HECC are shown in Figure 3.

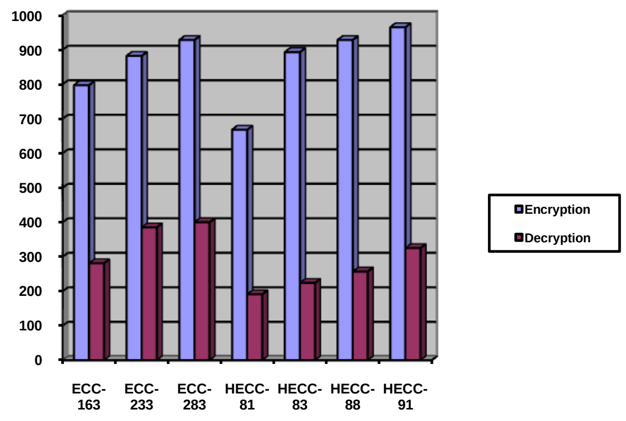

Fig-3 – Comparison of Encryption and Decryption time of ECC and HECC

Our results show that

- Both ECC and HECC take more time in encryption process compared to decryption process.
- Both ECC and HECC encryption and decryption time increases in higher field orders compared to lower field orders.
- HECC decryption time is relatively less compared to ECC decryption time for same security level.

5. CONCLUSION

Curve based Cryptosystem like ECC and HECC are extensively used for all kinds of embedded processor architectures, where resources such as storage, time or power are constrained. This paper explores in details main operations like scalar multiplication, group operations etc which are the prime steps for efficient implementation of ECC/ HECC. We have implemented ECC (genus 1) and HECC (genus 2) on different binary fields. We observe that genus 2 HECC is faster than ECC in the experiment to study the relative performance of elliptic curve and hyperelliptic curves cryptosystem. In our view, genus 2 HECC has the advantage over ECC in constrained environment due to its short operand size and it takes less processing time for basic operations like encryption, decryption.

6. REFERENCES

[1] Koblitz, N. 1987. "Elliptic curve cryptosystems", Mathematics of Computation 48, pp.203–209.

[2] Koblitz, N. 1989. "Hyperelliptic cryptosystems", Journal of Cryptology 1,3, pp.139–150.

[3] Darrel Hankerson, Julio Lopez Hernandez, Alfred Menezes, 2000. "Software Implementation of Elliptic Curve Cryptography Over Binary Fields", Cryptographic Hardware and Embedded Systems- CHES, LNCS vol 1965, Springer- Verlag, pp.243-267.

[4] Darrel Hankerson, Alfred Menezes, Scott Vanstone, 2004. "Guide to Elliptic Curve Cryptography" Springer publication, ISBN 0-387-95273-X.

[5] Menezes A, Wu Y, Zuccherato R, 1997. "An elementary introduction to hyperelliptic curves", available at http://www.cacr.math.uwaterloo.ca/techreports/1997/tech-reports97.html.

[6] J.Pelzl, T.Wollinger, J.Guajardo and C.Paar, 2003. "Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves", Cryptology ePrint Archieve, Report 026, http://eprint.iacr.org/, pp.351-365.

[7] P.Longa, C.Gebotys, 2010. "Efficient techniques for High Speed Elliptic Curve Cryptography" CHES, LNCS 6225, pp:80-94.

[8] Patrick Longa, Catherine Gebotys, 2009. "Novel Precomputation Schemes for Elliptic Curve Cryptosystems", ACNS, LNCS vol-5536, pp:71-88.

[9] Patrick Longa, Ali Miri, 2008. "Fast and Flexible Elliptic Curve Point Arithmetic over Prime Fields", IEEE Transactions on Computers, vol. 57, no. 3, pp. 289-302,

[10] Kakali Chatterjee, Daya Gupta, 2009. "Secure access of smart cards using Elliptic Curve Cryptosystems", WiCOM, IEEE Catalog No.CFP09WNM-CDR

[11] Henry Cohen and Gerhard Frey, 2006. "Handbook of Elliptic and Hyperelliptic Curve Cryptography", Chapman & Hall/CRC Press.

[12] Tanja Lange, 2005. "Formulae for Arithmetic on Genus 2 Hyperelliptic Curves" Applicable Algebra in Engineering, Communication and Computing, Volume 15, Number 5,

295-328, DOI: 10.1007/s00200-004-0154-8 http://www.itsc.ruhr-uni-bochum.de/tanja.

[13] T. Lange. 2002. “Efficient Arithmetic on Genus 2 Hyperelliptic Curves over Finite Fields via Explicit Formulae”, Cryptology ePrint Archive, Report 2002/121, http://eprint.iacr.org/.

[14] Roberto Maria Avanzi, 2003. “Aspects of hyperelliptic Curves over Large Prime Fields in Software Implementation”, http://www.arehcc.com.

[15] J.Pelzl, T.Wollinger, C.Paar, 2004. “High performance arithmetic for Hyper elliptic curve cryptosystem of Genus 2”, International Conference on Information Technology: Coding and Computing (ITCC) Volume 2.

[16] J.Pelzl, T.Wollinger, C.Paar, 2003. “Elliptic & Hyperelliptic Curves on Embedded µP”, ACM special issue Security and Embedded Systems Vol.no.0164-0925/99/0100-0111.

[17] Kakali Chatterjee, Daya Gupta, 2010. “Evolution of Hyperelliptic Curve Cryptosystems”, in proceedings of ICDCIT, LNCS 5966, Springer -Verlag Berlin Heidelberg, pp.206-211

[18] Cantor D G. 1987. “Computing in the Jacobian of a hyperelliptic curve”, Mathematics of Computation, 48: pp.95-101.

[19] R.Harly, 2000. “Fast Arithmetic on Genus Two Curves”, available at http://cristal.inria.fr/”harly/hyper.