Performance Study of genus 3 Hyperelliptic Curve Cryptosystem

Daya Gupta*, Asok De** and Kakali Chatterjee*

Abstract—Hyperelliptic Curve Cryptosystem (HECC) is well suited for all kinds of embedded processor architectures, where resources such as storage, time, or power are constrained due to short operand sizes. We can construct genus 3 HECC on 54-bit finite fields in order to achieve the same security level as 160-bit ECC or 1024-bit RSA due to the algebraic structure of Hyperelliptic Curve. This paper explores various possible attacks to the discrete logarithm in the Jacobian of a Hyperelliptic Curve (HEC) and addition and doubling of the divisor using explicit formula to speed up the scalar multiplication. Our aim is to develop a cryptosystem that can sign and authenticate documents and encrypt / decrypt messages efficiently for constrained devices in wireless networks. The performance of our proposed cryptosystem is comparable with that of ECC and the security analysis shows that it can resist the major attacks in wireless networks.

Keywords— Hyperelliptic Curve Cryptosystem(HECC), Secure Hyperelliptic Curve, Hyperelliptic Curve Deffie-Hellman(HECDH), Hyperelliptic Curve Digital Signature Algorithm (HECDSA)

1. INTRODUCTION

Public Key Cryptography plays an essential role in wireless network as these networks are vulnerable to both active and passive attacks. Security mechanisms are essential to ensure the integrity, confidentiality, and authenticity of the data that are transmitted in such networks. A lot of information can be revealed online due to various attacks like forgery, impersonation attack, insertion attack, etc. Therefore, to protect the information between the user (client) and information provider (server), the message must be transmitted in an encrypted way. If an encrypted message is transmitted, the cryptographic framework has to ensure that no other party can obtain any information about the message or change it without being noticed. To fulfill this need, a digital signature is necessary which helps to guarantee the reliability, non-repudiation and unforgeability. Most of the systems use RSA based digital signatures [1, 2]. However RSA signature is not suitable for constrained devices as it requires long key length. This approach can lead to a number of problems such as increased processing time (decryption time increases about 8 times as key sizes double) and increased key storage requirement (for private and public key certificates). Cryptosystems based on Elliptic Curve and Hyperelliptic Curve are considered to

Manuscript received August 24, 2011; first revision November 30, 2011; accepted December 23, 2011.
Corresponding Author: Kakali Chatterjee
* Computer Engineering Department, Delhi Technological University, Bawana Road, Delhi 110042, India (daya_gupta2005 @yahoo.co.in, kakali2008@gmail.com)
** Ambedkar Institute of Advanced Communication Technologies & Research, Delhi 110031, India (asok.de@ gmail.com)

be suitable for platforms with limited resources since they require smaller fields than RSA to attain the same security level. The purpose of this paper is the performance study of an alternative technology, namely HECC, which is based on curve arithmetic and offer significant benefits over RSA and ECC when used in constraint devices in wireless network.

To provide secure communication in a wireless network, authenticated key agreement protocol is an important primitive for establishing session key. Existing authentication protocol [3] certify users through a third party called Certification Authority who issues public keys for secure communication. But this increases the traffic a lot by introducing frequent certificates, which results in higher energy consumption and also key administration overhead. To overcome these problems, many key exchange protocols like PKE [4], EPA [5], and SKA [6] are introduced. Most of these protocols are password based where a directed dictionary attack can almost always succeed to break the password. To overcome this problem, ECC based authenticated key agreement protocol in wireless network are discussed in [7-10]. These protocols utilize ECDSA signature technique which enhances the security level of user authentication and key exchange. However, the security level can also be increased using hyperelliptic curve because it has some advantage over ECC. For instance, in ECC we have to work with operand lengths of approximately 160-bit whereas in the case of HECC, one needs 40-bit to 80-bit long operands to compute the group operations for these curves. Thus, HECC is more suitable for implementation in the constrained platforms like the PDA, smartcard, and handheld devices etc. in wireless network.

Current research on HECC emphasizes finding efficient methods to select secure hyperelliptic curves, fast operations on the Jacobians, and implementation of HECC for use in practical applications to enhance network security. In this direction, we have explored various possible attacks to the discrete logarithm in the Jacobian of a hyperelliptic curve that are to be considered to establish a secure HEC. Then we explore the group operations on a Jacobian in detail so as to obtain the explicit formula for performing addition and doubling in an efficient way to speed-up the arithmetic on genus-3 hyperelliptic curves.

Our contributions in this paper are as follows:

i) We have implemented Hyperelliptic Curve Diffie-Hellman (HECDH) key agreement protocol for secret key generation and Hyperelliptic Curve Digital Signature Algorithm (HECDSA) for signature generation /verification, considering the genus 3 Hyperelliptic Curve $C: v^2 + (u^2 + u)v = u^7 + u^5 + u^4 + u^3 + u^2 + u + 1$ over the finite field $\mathbb{F}_2{}^7$ using Netbeans IDE 6.8.
We have preferred Java over C++, as Java enables the development of robust applications on multiple platforms in heterogeneous distributed networks.

ii) The security analysis of the proposed protocol shows that it can resist the main attacks from both internal users and external hackers.

The rest of the paper is organized as follows:

In Section 2, Secure Hyperelliptic Curve is discussed; Section 3 provides Mathematical Background; Section 4 presents proposed Hyperelliptic Curve Cryptosystem; Section 5 presents Implementation Results; Section 6 presents Security Analysis of the proposed cryptosystem. Finally, we conclude the paper in Section 7.

2. SECURE HYPERELLIPTIC CURVE

Hyperelliptic Curves present a rich source of abelian groups over which the discrete logarithm problem is believed to be difficult. Hence these groups can be used for the implementation of various public key primitives. If the curves are chosen carefully then the DLP in these groups is as hard as for general groups and one can use much smaller parameters and key sizes than when using the multiplicative group of finite fields and still obtain the same level of security. The selection of a secure hyperelliptic curve is explored in this section:

Let a hyperelliptic curve C of genus g is defined on a finite field $\mathbb{F}_q$ ($q = p^r$ and p is a prime), and given by the equation $C: y^2 + h(x)y = f(x)$. The order of the Jacobian $\mathbb{J}(\mathbb{F}_q)$ of C, denoted by $\# \mathbb{J}(C; \mathbb{F}_q)$, should be divisible by a large prime number l of at least 40 decimal digits [11].

Let $\mathbb{F}_q^n$ denote the degree n extension of $\mathbb{F}_q$. Its Jacobian $\mathbb{J}(C; \mathbb{F}_q^n)$ over $\mathbb{F}_q^n$ is a finite abelian group and $(q^{n/2}-1)^{2g} \leq \#\mathbb{J}(C; \mathbb{F}_q^n) \leq (q^{n/2}+1)^{2g}$. The HCDLP in $\mathbb{J}(C; \mathbb{F}_q^n)$ is: given two divisors D_1, D_2 defined on $\mathbb{J}(C; \mathbb{F}_q^n)$ over $\mathbb{F}_q^n$, to determine the integer m such that $D_2=mD_1$, provided that such an integer m exists.

To establish a secure HEC, we should select the hyperelliptic curve so that its Jacobian satisfies the following conditions:

1) There is an index calculus attack on $\mathbb{J}(C; \mathbb{F}_q^n)$ that is more efficient than Pollard's rho method if the genus g of C is not small enough. Initially, this attack was developed for high genus curves by Adleman, Demarrais, and Huang [12]. They found a sub-exponent time algorithm to solve the DL in the Jacobian of hyperelliptic curves of a big genus over a finite field. Curves of higher genera (preferably $g\leq4$) are, therefore, not suitable for cryptographic use ($2g+1 < \log q^n$).
2) Any naive implementation that neglects security aspects like selection of the curve can be broken by the generic cryptanalytic attacks. If the group order is large, but is divisible by only small primes, the DLP can be broken by Pohlig-Hellman attack. Therefore, $\#\mathbb{J}(C; \mathbb{F}_q^n)$ should have a large prime factor so as to prevent the attacks of Pohlig-Hellman's methods. Since the time complexity of Pohlig-Hellman's method is proportional to the square root of the largest prime factor of $\#\mathbb{J}(C; \mathbb{F}_q^n)$, so far it is demanded that this largest prime factor should be at least 160-bit in length.
3) In order to prevent the attack of Frey [13], which uses the Tate pairing generation of MOV attacks, the large prime factor of $\# \mathbb{J}(C; \mathbb{F}_q^n)$ should not divide $(q^n)^k - 1$, here $k < (\log q^n)^2$.
4) In order to prevent the attack generated by Ruck [14], the Jacobian of a hyperelliptic curve over the large prime field GF(p) should not have p-order subgroup.
5) Simple side channel attacks obtain information from a single scalar multiplication by observing leaked information [15]. For restricted devices like smartcards it is possible for an attacker to derive side-channel information on the operations performed. To harden a cryptographic primitive against simple side-channel attacks, we make the observable information independent of the secret scalar. This is achieved by one of the following three approaches: inserting dummy arithmetic instructions, using indistinguishable / unified addition and doubling formulas, or applying Montgomery's ladder for scalar multiplication.

3. Mathematical Background

Hyperelliptic Curve Cryptosystem (HECC) was proposed by Koblitz [11], based on the discrete logarithm problem on the Jacobian of hyperelliptic curves over finite fields.

3.1 Arithmetic of Hyperelliptic Curve

A hyperelliptic curve C of genus g over $\mathbb{F}_q$ is an absolutely irreducible non-singular curve defined by $C : y^2 + h(x)y = f(x)$, where $h, f \in \mathbb{F}_q$ [x], are such that $y^2 + h(x)y - f(x)$ is absolutely irreducible over $\mathbb{F}_q$, and if $b^2 + h(a)b = f(a)$, for (a,b) $\in$ $\overline{\mathbb{F}} \times \overline{\mathbb{F}}$, then $2b + h(a) \neq 0$ or $h'(a)\,b - f'(a) \neq 0$ [16].

Definition 1- A hyperelliptic curve C is called an imaginary curve if q is odd, then f is monic, deg $(f) = 2g+1$ and $h = 0$. If q is even, then h and f are monic, $deg(f) = 2g + 1$ and $deg\ (h) \leq g$.

Definition 2- A hyperelliptic curve C is called a real curve if q is odd, then f is monic, deg $(f) = 2g+2$ and $h = 0$. If q is even, then h is monic , $\deg(h) = g+1$ and $\deg(f) \leq 2g+1$ or $deg(f) = 2g+2$ and the leading coefficient of f is of the form $\beta^2 + \beta$ for some $\beta \in \mathbb{F}_q$.

Definition 3- A hyperelliptic curve C is called an unusual curve if $\mathbb{F}_q$ has odd characteristic, then $deg\ (f) = 2g+2$ and if $\mathbb{F}_q$ has characteristic 2, then deg $(h) = g+1$ and $deg\ (f) = 2g+2$ and the leading coefficient of f is not of the form $\beta^2 + \beta$ for some $\beta \in \mathbb{F}_q$

Consider a Hyperelliptic curve C as defined by $C: y^2 + h(x)y = f(x)$. 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 $\mathbb{D}$(C). The set of degree zero divisors $\mathbb{D}^0$ forms a subgroup of $\mathbb{D}$(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 $\mathbb{P}$. If $D_1, D_2 \in \mathbb{D}^0$ then we write $D_1 \sim D_2$ if $D_1 - D_2 \in \mathbb{P}$; D_1 and D_2 are said to be equivalent divisors. Now, we can define the Jacobian of C as the quotient group $\mathbb{D}^0/\mathbb{P}$ [17].

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 $\mathbb{F}$ [16].

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 [18]. 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 to $\mathbb{J}_C$.

For a genus 3 hyperelliptic curve C over $\mathbb{F}_q$ defined as C: $Y^2 = \mathbb{F}(X)$, a semi-reduced divisor can be represented by the following pair of polynomials :

(**Mumford's representation**)

D = (U, V), U,V $\in \mathbb{F}_q[X]$; where

$U = \prod (X - x_i)^{ordpi\ (D)}$,

$y_i = V(x_i)$

for $P_i = (x_i,y_i) \in C$ with $\mathrm{ord}_{pi}(D) > 0$, and
deg $V <$ deg U, $F - V^2 \equiv 0 \bmod U$.

The degree of U is called the weight of D, and D is a reduced divisor, if its weight equals 3. Any class in $J_C(\mathbb{F}_q)$ is uniquely represented by a reduced divisor (i.e., each class includes a unique reduced divisor).

Unlike elliptic curve, 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. The group elements have a nice cannonical representation by means of two polynomials of small degree. The basic algorithm of performing arithmetic for divisor addition and doubling in the Jacobian of hyperelliptic curve is Cantor's algorithms.

3.2 Group Operations on a Jacobian

Group operations of divisor on $\mathbb{J}_c(\mathbb{F})$ are performed in the following two steps: addition of generic divisors and doubling of generic divisors. Addition of divisor classes means multiplication of ideal classes, which consists of a composition of the ideals and a first reduction to a basis of two polynomials. The output of this algorithm is called semi-reduced divisor. Then the second algorithm (reduction) is used to find the unique representative in the class. Cantor's algorithm [18] is used for transferring the group laws in a sequence of composition and reduction using only polynomial arithmetic.

```
Cantor's Algorithm for Group Addition
Input: $D_1 = [U_1, V_1]$ and $D_2 = [U_2, V_2]$ , $C: Y^2 = F(X)$
Output: $D_3=(U_3,V_3)$ reduced with $D_3 = D_1 + D_2$
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 $s_1 = c_1e_1, s_2 = c_1e_2, s_3 = c_2$;
4. $U' = u_1u_2/d^2$;
5. $V' = \{s_1u_1v_2+s_2u_2v_1+s_3(v_1v_2+f)\}/d \bmod u$
6. $U_3 = (F-v^2) / U'$, $V_3 = - V' \bmod U_3$
7. Make $U_3$ monic.
```

Harley's algorithm for genus 3 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 a 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. Subsequently, Harley's algorithm was used for fast arithmetic on genus 3 HEC.

First classification of the input divisor classes is done by using the weights of them and then another classification of the divisor classes is done by testing $\gcd(U_1,U_2) = 1$ for addition $D_3 = D_1 + D_2$, $D_1 = (U_1,V_1)$, $D_2 = (U_2, V_2)$ or by testing $\gcd(U_1,2V_1) = 1$ for doubling $D_2 = 2D_1$, $D_1 = (U_1, V_1)$. These gcd computations are carried out by a resultant computation. The case satisfied deg U_1 = deg U_2 = 3 and $\gcd(U_1,U_2) = 1$ for addition and the case satisfied deg $U_1 = 3$ and $\gcd(U_1,2V_1) = 1$ for doubling, are called the most frequent cases.

3.3 Explicit Formula for genus 3 Curves

The first explicit formula for genus 2 proposed by Harley [19] has been followed by the work of Lange [20, 21]. After extensive research on explicit formula for performing addition and doubling, Avanzi [22] proposes a software implementation of genus 2 and genus 3 hyperelliptic curves over large prime fields. Pelzl and Wollinger [17, 23] propose a cost effective explicit formula for genus 2 and genus 3 curves and give the first implementation of a HEC cryptosystem on an embedded processor. Gonda et al. [24] propose improvements of addition algorithm on genus 3 HEC and implemented it on a 64-bit CPU. Fan et al. [25] discussed the performance of genus 3 HECC over three different binary fields. The idea to use HEC for cryptographic applications has been further analyzed and implemented in software and hardware oriented platforms by Kuroki et al. [26], Sakai and Sakurai [27], Nagao [28], and Smith [29]. We discussed the evolution of Hyperelliptic Curve Cryptosystems in [30].

Explicit Formula for addition and doubling of divisors based on [17, 23, and 25] is discussed below. The most frequent input for addition consists of two divisor classes represented by $[U_1,V_1]$, $[U_2,V_2]$, where $\deg(U_1) = \deg(U_2) = 3$ and $\gcd(U_1,U_2) = 1$. This guarantees that the associated reduced divisors D_1, D_2 do not have any point or its opposite in common and both divisors have 3 affine points in the support. For the doubling it is assumed that the class is represented by $[U_1, V_1]$ with $\deg(U_1) = 3$ and that $\gcd(U_1, 2V_1) = 1$.

Explicit Formula for addition of divisors of genus 3 HEC (most frequent cases)
Input:- C: $Y^2 = F(x)$, $F=x^7+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$;
Reduced divisors $D_1= (U_1,V_1)$ and $D_2= (U_2,V_2)$
$U_1=x^3+u_{12}x^2+u_{11}x+u_{10}$, $V_1=v_{12}x^2+v_{11}x+v_{10}$, $U_2=x^3+u_{22}x^2+u_{21}x+u_{20}$, $V_2= v_{22}x^2+v_{21}x+v_{20}$
Output:- Reduced divisor $D_3=(U_3,V_3)=D_1+D_2$
$U_3 = x^3+u_{32}x^2+u_{31}x+u_{30}$, $V_3= v_{32}x^2+ v_{31}x+v_{30}$

```
Steps:- 1. Compute Resultant r of $U_1$ and $U_2$. [using Bezout's theorem]
  If r=0 then call the Cantor Algorithm.
2. Compute pseudo inverse $I=i_2x^2+i_1x+i_0 \equiv r/U_1 \bmod U_2$
3. Compute $S'=s'_2x^2+s'_1x+s_0' = rS \equiv (V_2-V_1)I \bmod U_2$ [using Karatsuba multiplication]
4. Compute $S= (S'/r)$ and make S monic. [using Montgomery trick]
5. Compute $Z=x^5+z_4x^4+z_3x^3+z_2x^2+z_1x+z_0=SU_1$[using Karatsuba multiplication]
6. Computing $U_t = x^4+u_{t3}x^3+u_{t2}x^2+u_{t1}x+u_{t0} = (S(Z+2w_iV_1) - w_i^2((F-V_1^2)/U_1))/U_2$
   [using karatsuba multiplication, Efficient Division]
7. Compute $V_t = v_{t3}x^3 + v_{t2}x^2+v_{t1}x+v_{t0} \equiv - (wZ+ V_1) \bmod U_t$
8. Compute $U_3=x^3+u_{32}x^2+u_{31}x+u_{30} = (F-V_t^2)/U_t$ [using Efficient Division]
9. Compute $V_3=v_{32}x^2+v_{31}x+v_{30} \equiv - V_t \bmod U_3$
```

Explicit Formula for doubling of divisor of genus 3 HEC (most frequent cases)
Input:- C: $Y^2 =F(x)$, $F= x^7+f_5x^5+f_4x^4+f_3x^3+f_2x^2+f_1x+f_0$;
Reduced Divisor $D_1= (U_1, V_1)$, $U_1 = x^3+u_{12}x^2+u_{11}x+u_{10}$, $V_1= v_{12}x^2+v_{11}x+v_{10}$
Output:- Reduced divisor $D_2=(U_2,V_2)=2D_1$, $U_2= x^3+u_{22}x^2+u_{21}x+u_{20}$, $V_2= v_{22}x^2+v_{21}x+v_{20}$

```
Steps:- 1. Compute Resultant r of $U_1$ and $2V_1$. [using Bezout's theorem]
  If r=0 then call the Cantor Algorithm.
2. Compute pseudo inverse $I = i_2x^2+i_1x+i_0 \equiv r/(2V_1) \bmod U_1$
```

3. Compute $Z=z_2x^2+z_1x+z_0=((F-V_1^2)/U_1)\mathrm{mod}U_1$ [using Efficient Division]
4. Compute $S' = s_2'x^2+s_1'x+s_0' = rS \equiv Z\ I\ \mathrm{mod}U_1$ [using Karatsuba multiplication]
5. Compute $S=(S'/r)$ and make S monic [using Montgomery trick]
6. Compute $G=x^5+g_4x^4+g_3x^3+g_2x^2+g_1x+g_0=SU_1$ [using Karatsuba multiplication]
7. Compute $U_t = x^4+u_{t3}x^3+u_{t2}x^2+u_{t1}x+u_{t0} = ((G+w_iV_1)^2-w_i^2F)/U_1^2$
8. Compute $V_t= v_{t3}x^3+v_{t2}x^2+v_{t1}x+v_{t0}\equiv -(wG+V_1) \bmod U_t$
9. Compute $U_2= x^3+u_{22}x^2+u_{21}x+u_{20} = (F-V_t^2)/U_t$ [using Efficient Division]
10. Compute $V_2=v_{22}x^2+v_{21}x+v_{20} \equiv -V_t\ \mathrm{mod}U_2$

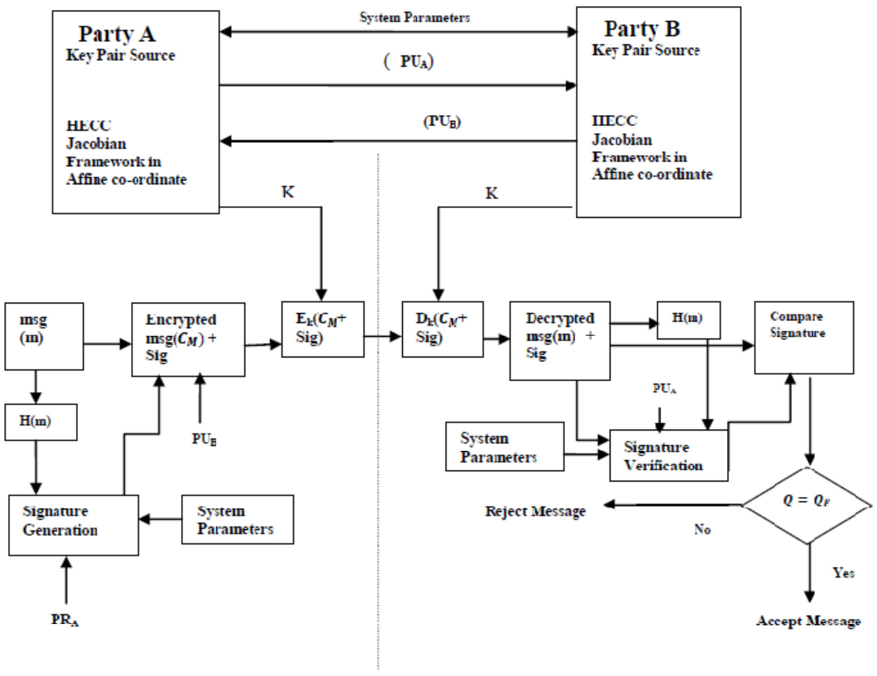
In the above case the cost of addition and doubling of divisor is I+68M+12S and I+60M+16S respectively where I= Inversion, M = Multiplication, S= Squaring.

4. Proposed Hyperelliptic Curve Cryptosystem

The proposed cryptosystem is shown in Fig.1.

The system works as follows:

Party A (Client) and Party B (Server) will generate their private keys (PR_A, PR_B) and public keys (PU_A, PU_B) using HEC. Before generating key pairs, the system parameters hyperelliptic curve C, prime p, divisor D, and elements of the group G of order N are exchanged in the initialization phase. For authentication, Party A first takes message m. The message is input to a hash function that produces a secure hash code of a fixed length. The hash code is provided as



input to a signature function along with a random number k generated by PRNG (Pseudo Random Number Generator) for this particular signature. The function also depends on PR_A and system parameters ($\mathbb{F}_2^n$, C , D, N, c, p). The result is a signature consisting of two components, Q and S. After that the message is first encrypted and binded with the signature. Then the block containing the encrypted message (C_M) and signature (Sig) is again encrypted with the common secret key (K) and send it to the Party B.

At the receiving end, Party B first decrypts the block containing the message and the signature with the common secret key K. Now Party B gets the encrypted message and signature separately. Again Party B will decrypt the encrypted message (C_M) and a hash code of the incoming message is generated. The incoming message and the signature is the input to a verification function. The verification function also depends on the PU_A, system parameters ($\mathbb{F}_2^n$,C,D,N,c, p). The output of the verification function is a value that is compared with the signature component. If the condition ($Q = Q_F$) is satisfied then the signature is considered to be a valid signature and the message is accepted. We divide the proposed protocol into four different phases. Our proposed protocol uses modified HECDH and HECDSA to enhance security.

The phases of our proposed protocol are described below:

Initialization Phase

During this phase, Party A first calculates X= h(CIN) where CIN is the Client Identification No. and h( ) is a one way hash function. Now the client (Party A) generates a pre-knowledge message containing the client network identity (ID), X, and system parameters (hyperelliptic curve C, prime p and divisor D, representation of field elements of order N, and P a point on the curve) and transmit it to the server (Party B). Once the pre-knowledge message is received, server (Party B) will match the received value of X with the stored value in the verification table (this table is generated after the deployment of the network). If the values are same, then the server (Party B) agrees for further communication and sends an acknowledgement of the pre-knowledge message to Party A. After that both parties will generate a common secret key for secure communication.

Secret Key Generation Phase

In this phase we use Hyperelliptic Deffie-Hellman key exchange protocol for establishing a common secret key by applying the following steps:

► Party A will generate Private key $PR_A \in_R N$ [choose a prime (PR_A) at random in N].

► Party A will generate Public key $PU_A = [PR_A]\, D$ and send PU_A to Party B.
[PU_A is represented using Mumford representation which is of the form $(u(x);\ v(x))$];

► Similarly Party B will generate the private key PR_B, public key $PU_B = [PR_B]D$. Party B will send the public key PU_B Party A.

► Once Party A and Party B exchange their public keys, Party A computes $M_1 = [PR_A]$ P and common secret key $K = (PR_A + M_1)(PU_B)$
Here P is a point on the curve whose x co-ordinate is considered for calculation of M_1, M_2.

► On the other side Party B will computes $M_2 = [PR_B]$ P and generates the common secret key $K = (PR_B + M_2)(PU_A)$. If the protocol works correctly, both the Party A and Party B generates the same value of K. This can be proved by the simple mathematical calculation shown below:

$$K = (PR_A + M_1)(PU_B) = (PR_A + M_1) [PR_B]D = [PR_A][PR_B]D + [PR_A] \text{ P } [PR_B]D = [PR_A]D \, [PR_B] + [PR_A]D \, [PR_B] \text{ P} = PU_A(PR_B + M_2)$$

Signature Generation and Verification Phase

HECDSA signature scheme is used here. For generation of signature we apply hash functions to message m. Here, we assume that E∈G is represented using Mumford representation which is of the form E = $[u_E, v_E]$ with $u_E = x^e + \sum_{i=0}^{e-1} u_i$ where $e \leq g$

- ► A generates a random nonce k $\in_R$ N and produce $E \leftarrow [k]D$ where D is each non-trival group element represented using Mumford representation.
- ► Party A calculates Q = $\sum_{i=0}^{e-1} L(u_i)q^i \bmod p$ where e is an integer with $e \leq g$ and assuming the finite field elements are ordered such that $0 \leq L(u_i) < q$
- ► $S \leftarrow (k^{-1}(h(m) - [PR_A]Q)) \bmod l$ where l =N/c (c is co-factor)
- ► The signature (Q, S) then binds with the message to provide authentication and send it to Party B.

For verification Party B follow the steps as described below:

- ► If Q or $S \notin [1, l-1]$ then reject the signature
- ► Else Party B generate $w \leftarrow S^{-1} mod\ l$ and R_1 and R_2 such that
- ► $R_1 \leftarrow [h(m)w] \ mod\ l$
- ► $R_2 \leftarrow [Qw] \ mod\ l$
- ► $F \leftarrow [R_1] \, D \oplus [R_2] \, PU_A$ where F is also in the form $F = [u_F, v_F]$
- ► If $F = 0$ then reject else calculate $Q_F = \sum_{i=0}^{e-1} L(u_{F,i})q^i \bmod p$
- ► If $Q = Q_F$ then accept the signature else reject.

Encryption and Decryption Phase

For message encryption and decryption, we follow the methods of HECC encryption and decryption [15].

In the encryption process first message m will be encoded as a series of points which is represented as $(u(x), v(x))$ noted as E_M. To encrypt this message, Party A will perform the following steps:

- ► Party A generates $W \leftarrow [k]D$ [W is in the form $(u(x); v(x))$] where k is previously generated random prime no. and D is the divisor.
- ► Produce the cipher text $C_M \leftarrow \{W, E_M + [k]PU_B\}$ which is sent to Party B.

To decrypt the cipher text C_M, Party B multiplies the first part in the pair by Party B's private key PR_B and subtracts the result from the second part of the pair. The original message can be retrieved from the cipher text as shown below:

$$E_M + [k]PU_B - PR_B\,(W) = E_M + [k]PU_B - PR_B\,[k]D = E_M + [k]PU_B - [k](PR_B\,D)$$
$$= E_M + [k]PU_B - [k]PU_B = E_M$$

5. Implementation Results

We have implemented genus 3 HECC on different binary fields using Netbeans IDE 6.8. Netbeans refers to both a platform framework for Java desktop application and an Integrated

Table 1. Experimental Results of genus 3 HECC (Binary Field)

G-3 Binary field	**Curve (C) Equation**	**Divisor Generation**	**Public Key** PU$_A$	**Public Key** PU$_B$	**Party-A Secret key K**	**Party-B Secret key K**	**Signature Generation (Party-A)**	**File encryption (Party-A)**	**File decryption (Party-A)**	**Signature Verification (Party-A)**
Field Order-F(2^56) Group Order-(2^168)	C : v^2 + (u^2 + u)v = u^7 + u^5 + u^4 + u^3 + u^2 + u + 1 (2 ms)	D$_1$: div (u^3 + 12u^2 + 12u + 1, u^2 + 2u + 8) D$_2$: div (u^3 + u^2 + 13u +12, 12u^2+fu+ e) D=D$_1$+D$_2$: div (24u^3 + 1fu^2 + 1fu + f, 11u^2 + 1au + 13) Dinv: div (u^3 + 12u^2 + 12u + 1, 3u + 8) (6 ms)	div (u^3 + 14u^2 + u + 1f, fu^2 + u + 1) (14 ms)	div (u^3 + 16u^2 + 11, 7u^2 + 15u + 1d) (15 ms)	div (24u^3 + u^2 + 18u + 6, 6u^2 + 12u + 12) (14 ms)	div (24u^3 + u^2 + 18u + 6, 6u^2 + 12u + 12) (15 ms)	Signature = [AC0CFQCSOUYUNUJHldm0utAF6aCB5z,rawIUcQ6WYQJukWovro9uwY59gHGK5] Time in ms: 60	Encryption done : plain.txt.enc (580ms)	Decryption done : plain.txt.enc.txt (150 ms)	Signature verification successful! Time in ms: 30
Field Order-F(2^54) Group Order-(2^162)	C: v^2 + (u^2 + u)v = u^7 + u^5 + u^4 + u^3 + u^2 + u + 1 (2 ms)	D$_1$: div (u^3 + 11u^2 + 12u + 1, u^2 + 2u + 6) D$_2$: div (u^3 + u^2 + 13u + 12, 12u^2 + fu + e) D=D$_1$+D$_2$: div (u^3 + 1du^2 + 8u + 1, 10u^2 + 11u + 5) Dinv: div (u^3 + 11u^2 + 12u + 1, 3u + 6) (7 ms)	div (u^3 + 18u^2 + 1au + 12, au^2 + 4u + 1a) (16 ms)	div (24u^3 + 18u^2 + 19u + 15, 3u^2 + 17u + 8) (17 ms)	div (u^3 + 17u^2 + 12u + 4, 2u^2 + 7) (16 ms)	div (u^3 + 17u^2 + 12u + 4, 2u^2 + 7) (17 ms)	Signature = [FBACFEw4DEOhMZY5PBa8ertdfgEtDO,ym3JAhRbIF8OQIq6bfbYDszvZ4i4yCsg3Fw] Time in ms: 53	Encryption done : plain.txt.enc (682 ms)	Decryption done : plain.txt.enc.txt (180 ms)	Signature verification successful! Time in ms: 31

Development Environment (IDE) for developing mobile and web applications with Java. In the Java architecture, the security API (java.security package) for the Java Development Kit (JDK) introduced the Java Cryptography Architecture (JCA), which allows the generation of digital signature and message digests and more specifically Java Cryptography Extension (JCE), which provides implementation for key generation and agreement, encryption, and decryption algorithm.

We have considered the hyperelliptic curve $C: v^2 + (u^2 + u)v = u^7 + u^5 + u^4 + u^3 + u^2 + u + 1$ of genus 3 over the finite field $\mathbb{F}_{2^7}$. P=(α^{30} , 0) is an ordinary point in C($\mathbb{F}_{2^7}$). A divisor D can be represented as $D = div(a, b)$ with $a, b \in \mathbb{F}$, such that $D \sim D_1 + D_2$ where $D_1 = div(a_1, b_1)$, $D_2 = div(a_2, b_2)$. After computing the semi-reduced divisor $D = D_1 + D_2$ and $D' \sim D$, we have implemented Hyperelliptic Curve Digital Signature Algorithm in Netbeans IDE 6.8. The timings of basic operations using Netbeans IDE 6.8 are shown in Table 1 (after the first round). The timings have been measured on a PC with an Intel Core 2DUO CPU T6400@ 2.00GHz and windows vista operating system having jdk1.

5.1 Performance Analysis

i) We have compared our results with the timings of the basic operation found in [31]. These are 80 ms for secret key generation, 150 ms for signature generation, and 230 ms for signature verification with 163-bit key size based on ECC as against our result of 16 ms for secret key generation, 53 ms for signature generation, and 31 ms for signature verification

Table 2. Comparison of HECC with ECC for equivalent key sizes

Reference	Curve (key size)	Point Scalar Multiplication	Secret Key Encryption	Hash Function
Our result	HEC (54-bit key)	2	1	1
Lim et al. [10]	EC (160-bit key)	3	1	2
Aydos et al. [3]	EC (160-bit key)	3	2	1

with 54-bit key size based on HECC. Hence, our experimental result shows that the proposed cryptosystem is efficient as it takes less time for basic authentication operations.

ii) An efficient authentication protocol takes into consideration the communication and computation load during the user (client) authentication phase. The total number of bits exchanged in this protocol is 780 bits. The proposed protocol also has a low computation load on the client side compared to the existing protocols as shown in Table 2.

6. Security Analysis

In this section, we discuss the security of HECC. The proposed key agreement protocol will be considered to be a secure authenticated protocol if it satisfies the following properties:

Man-in-the-middle attack: This can be considered as an active attack. In this protocol, no useful information about the secret key K is revealed during a successful run. Consider A and B to be two communicating parties. Attacker I intercepts the public key PU_A and replaces it by PU_I. Then it sends PU_I to B and when B sends its public key PU_B, it again captures PU_B. But attacker I cannot compute the value of K and D because the security of an HEC is based on the difficulty of solving the discrete logarithm problem in the Jacobian of the curve. Thus attacker I cannot decrypt the useful messages or generate a valid signature during a successful run. Thus this protocol resists the man-in-the-middle attack.

Small subgroup attack: In a hyperelliptic cryptosystem, the system parameters ($\mathbb{F}_{2}{}^{n}$,C,D,N,c, p) are chosen in a manner such that the DL problem is too hard to compute. If hyperelliptic curve C has enough prime factors, the attacker could determine the secret scalar modulo of all these primes and recover a large part of the secret by using Chinese remaindering. This type of attack is called small subgroup attack. To avoid this attack, we check that D has order l where l is prime. For checking this, we first check that $[l\ ]D=0$ and computing $[h]D$ for $h=c/p_i$, for all prime divisors p_i of c and check that the result is not zero [15].

Known-key attack: In our proposed protocol, the client and the server both generate new PU_A and PU_B in every new session, and in addition the secret random no. k is changed with every new session also. Thus our proposed protocol is secure against known key attacks assuming that the hyperelliptic curve discrete logarithm problem is intractable.

Cipher text only attack: In a chosen ciphertext attack, the adversary knows the encryption algorithm and has access to many ciphertexts to be decrypted with an unknown key. In a chosen ciphertext attack the adversary uses the previous results to select subsequent ciphertexts with the secret key. In our proposed protocol, we never transmit the secret key K, which is used to encrypt the original encrypted message along with the signature. Even if an adversary knows the secret key, he cannot produce the original message as the cipher text consists of two parts, which is very difficult to decrypt as it lies on DLP in Jacobian of the curve.

Dictionary attack: In dictionary attack, the attacker pretends to be a legitimate client and attempts to login by guessing different passwords from a dictionary. As our protocol is not based on passwords, this type of attack is not applicable.

Perfect forward secrecy: In our protocol, perfect forward secrecy is maintained even if the user's public key is compromised since private key cannot be compromised as it is a random value that is changed from time to time. The adversary cannot decrypt the ciphertext as encryption is done using private key and public key, which depends on the difficulty of solving the discrete logarithm problem in the Jacobian of the curve. If we change the system parameters after each session then this problem becomes more hard. Thus, the property of perfect forward secrecy is satisfied.

7. Conclusion

HECC is a cryptosystem of choice when targeting embedded environments as the HEC operand size is only a fractional amount of the EC operand size and almost all the standard discrete logarithm based protocols such as the Digital Signature Algorithm (DSA) and EIGamal can be planted to HEC. We have explored in this paper various possible attacks that are to be considered to establish a secure HEC and efficient scalar multiplication. Our experimental results show that a public-key cryptosystem for constrained devices based on hyperelliptic curves can be designed to exchange keys, sign and authenticate documents, and encrypt and decrypt messages efficiently. In our view, HECC of genus 3 has the merit to be the preferred cryptosystem in constrained environment, as the performance of HECC with the operand size of 54-bit is comparable with the performance of ECC with an operand size of 160-bit.

References

[1] O.Goldreich, Y.Lindell, "*Session-Key Generation Using Human Passwords only," Crypto 2001, LNCS 2139*, pp.408-432.

[2] R.Katz, Q. Trovsky, M.Yang, *"Efficient Password Authenticated Key Exchange Using Human Memorable Passwords," Eurocrypt 2001, LNCS 2045*, pp.475-494.

[3] M.Aydos, T.Yanık, C.K.Koc, "*High-Speed Implementation of an ECC-based Wireless Authentication Protocol on an ARM Microprocessor", IEE Proceedings: Communications,* 2001, 148(5): pp.273-279.

[4] V. Boyko, P. Mackenzie, S. Patel. "*Provably secure password authenticated Key Exchange using Diffie-Hellman". EuroCrypt 2000, LNCS* pp.156-171.

[5] Y H Hwang, D H Yum, P J Lee, "*EPA: An Efficient Password-Based Protocol for Authenticated Key Exchange", ACISP 2003, LNCS 2727*, pp.452-463.

[6] E. Ryu, K. Kim, K. Yoo. "*A Simple Key Agreement Protocol", In Proc. of IEEE 37th Annual International Carnahan Conference 2003*, pp 128-131.

[7] K.Jung, J.Kim, T.Chung, "*Password-Based Independent Authentication and Key Exchange Protocol", ICICS-PCM 2003, IEEE*, pp.1908-1912.

[8] Julien Bringer, Hervé Chabanne and Thomas Icart, "*Password Based Key Exchange Protocols on Elliptic Curves Which Conceal the Public Parameters", ACNS 2010, LNCS 6123*, pp:291-308.

[9] Kakali Chatterjee, Asok De, Daya Gupta, "*Timestamp based Authentication Protocol for Smart Card using ECC", in proceedings of WISM 2011, LNCS 6987*, pp.368-375.

[10] Meng-Hui Lim, Chee-Min Yeoh, Sanggon Lee, Hyotaek Lim and Hoonjae Lee, "*A Secure and Efficient Three-Pass Authenticated Key Agreement Protocol Based on Elliptic Curves" NETWORKING 2008, LNCS 4982*, pp.170-182.

[11] Koblitz, N. 1989, "*Hyperelliptic cryptosystems", Journal of Cryptology 1,3*, pp.139-150.

[12] Adleman L, DeMarrais J,Huang M, "*A subexponential algorithm for discrete. logarithms over the rational subgroup of the Jacobians of large genus hyperelliptic curves over finite fields*", in ANTS-1, 1994, *LNCS 877*, pp.28-40.

[13] Frey G, Ruck H, "*A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves", Mathematics of Computation*, 1994, 62: pp 865-874.

[14] Ruck H.G "*On the discrete logarithms in the divisor class group of curves". Mathematics Computation*, 1999, 68: 805-806.

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

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

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

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

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

[20] Lange.T, "*Inversion-Free Arithmetic on Genus 2 Hyperelliptic Curves*", *Cryptology ePrint Archieve*, Report 147, 2002, http://eprint.iacr.org/.

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

[22] Roberto Maria Avanzi, "*Aspects of hyperelliptic Curves over Large Prime Fields in Software Implementation*", Dec 2003 available http://www.arehcc.com.

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

[24] Gonda.M, Matsuo.K, Kazumaro.A, Chao.J and Tsuji.S, "*Improvements of addition algorithm on genus 3 hyperelliptic curves and their implementations*", *Proc of SCIS* 2004, pp.89-96.

[25] Fan.X, Wollinger.T and Gong.G, "*Efficient explicit formulae for genus 3 hyperelliptic curve cryptosystems over binary fields", IET Inf.Secur.*, 2007,1,(2), pp.65-81.

[26] Kuroki.J, Gonda.M., Matsuo.K., Chao.J., Tsujii. S. 2002, "*Fast Genus Three Hyperelliptic Curve-Cryptosystems", SCIS* 2002, pp.503-507.

[27] Sakai.Y, and Sakurai, K., "*On the Practical Performance of Hyperelliptic Curve Cryptosystems in Software Implementation", in IEICE Trans.* Vol.E83-A NO.4, 2000, pp.692 – 703.

[28] Koh-ichi Nagao, "*Decomposed Attack for the Jacobian of a Hyperelliptic Curve over an Extension Field"*, 2007, http://eprint.iacr.org/ 2007/112.

[29] Benjamin Smith, "*Isogenies and the Discrete Logarithm Problem on Jacobians of Genus 3 Hyperelliptic Curves" EUROCRYPT* 2008, pp.163-180.

[30] Kakali Chatterjee, Daya Gupta, "*Evolution of Hyperelliptic Curve Cryptosystems", in proceedings of ICDCIT 2010, LNCS* 5966, pp.206-211.

[31] Nicholas Jansma, Brandon Arrendondo, "*Performance Comparison of Elliptic Curve and RSA Digital Signatures", University of Michigan*, 2004.

Daya Gupta
She is a Professor and the Head of the Computer Engineering Department at Delhi Technological University in India. She has received her Ph.D. in Computer Engineering from Delhi University. Her field of interest is Software Engineering, Information Security, etc. She has published many research papers in reputed international journals.

Asok De
He received his Ph.D. from IIT Kharagpur (India) and his field of interest is Microwave Antennas and Communication Systems. He is a Professor at the Delhi Technological University (formerly the Delhi College of Engineering). Presently he is working as the Principal at the Ambedkar Institute of Advanced Communications Technologies & Research in Delhi. He has published many research papers in reputed international journals.

Kakali Chatterjee
She is a Research Scholar in the Computer Engineering Department of Delhi Technological University (formerly the Delhi College of Engineering) in India. She has received her M.Tech (Information Technology) from the Centre for the Development of Advanced Computing, which is a R&D and academic centre of the govt. of India. Her field of interest is Information Security and Cryptography.