An Efficient Encryption Schemes based on Hyper-Elliptic Curve Cryptosystem

Md Hussain Ansari¹, Anup Kumar Keshri²

¹Assistant Professor, Department of Computer Science and Engineering, Guru Gobind Singh Educational Society’s Technical Campus, Bokaro Steel City, India
²Assistant Professor, Department of Information Technology, Birla Institute of Technology Mesra, Ranchi, India

Abstract—Encryption requires a better approach in terms of security and size. The main research concern of encryption in public key cryptography is Hyper-Elliptic Curve Cryptography. HECC uses less number of keys, that’s why it becomes more superior in security and efficiency, for example, a 60 bit HECC has the same security strength as 160 bit ECC & 1024 bit RSA. In this paper, we present an Encryption algorithm based on HECC techniques to calculate a point. The calculated point is now determine x and y coordinates by using HEC equation. This point is used for Encryption and Decryption with the help of shared key, which is obtained by Diffie-Hellman Key Exchange algorithm. The designing strategy of point not only reinforces the security and efficiency but also reduces the system overhead regarding size of keys.

Keywords—Public Key Cryptography, Elliptic Curve Cryptography, Hyper-Elliptic Curve Cryptography, Diffie-Hellman Key Exchange, RSA.

I. INTRODUCTION

In 1985 Neal Koblitz and Victor Miller proposed the Elliptic Curve Cryptosystem (ECC) [1]. The natural expansion of ECC (Elliptic Curves Cryptosystem) is HECC (Hyper-elliptic Curves Cryptosystem) which was first put forward by Neal Koblitz in 1988 [2]. Similar to the security of Elliptic Curve Cryptosystem, based on ECDLP (Elliptic Curves Discrete Logarithm Problem) [3,4], hyper-elliptic curves cryptosystem is based on the discrete logarithm problem of hyper-elliptic curves on finite field, i.e. HCDLP (Hyper-elliptic Curves Discrete Logarithm Problem). Hyper-elliptic curve cryptosystem is much superior to the Elliptic Curve Cryptosystem and RSA, such as high efficiency, short key length, and bandwidth [5-8].

Some researchers’ interests have recently fall on Hyper-Elliptic Curve because it offers equal security for a smaller key size. Since, HECC is a typically fast public key cryptosystem and it has superiority and more application efficiency. There are some superiorities of Hyper-Elliptic Curve cryptosystem such as high efficiency, short key length as compared with other public key cryptosystems and they are as follows:

- The security of hyper-elliptic curves cryptosystem is based on HECDL (Hyper-elliptic Curve Discrete Logarithm Problem) which is more superior than the jacobian group used in ECC.
- For hyper-elliptic curves cryptosystem with genus of 2, if the basic finite field is 60 bits and it is equivalent to 180 bits ECC, and in turn it is far more secure than 1024 bits RSA [9-11].
- Security of Hyper Elliptic Cryptosystem proves to be reliable because the attack algorithm against HECC with low genus g is inapplicable with exponent complexity.
- A relatively smaller basic field can be constructed by a secure Jacobian group with large prime number order in HECC [6,12].

II. THE MATHEMATICAL OVERVIEW

The HEC Cryptosystem makes use of Hyper-Elliptic curve, divisor, reduced divisor and adding divisor to calculate an efficient reduced point. In Hyper-Elliptic Curve coefficient are restricted to elements of a finite field [13].

A hyper elliptic curve H of genus g (g ≥ 2) over a field F is a nonsingular curve that is given by an equation of the following form:

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

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

\[ b(u)^2 + h(u)b(u) \equiv f(u)(\text{mod } a(u)). \]

Let H be a hyper-elliptic curve of genus g over a field F. A reduced divisor (defined over F) of H is defined as a form \( \text{div} (a(u), b(u)) \), where \( a(u), b(u) \in F[u] \) are polynomial.
If $D_1 = \text{div} (a_1, b_1)$ and $D_2 = \text{div} (a_2, b_2)$ are two reduced divisors defined over F, then we first add this two divisors with Cantor’s Algorithm and after this we find reduced divisor. From the Cantor’s Algorithm, we find the $D_3$ from these two divisors $D_1, D_2$ [14-15].

A high-genus Hyper-Elliptic Curve computationally simpler than low-genus curves because of a natural choice of small divisors that could be used in its index calculus. Namely elements of the Jacobian can be uniquely represented by the certain pairs of polynomials in the form $(a(x), b(x))$, where deg $a \leq g$ and deg $b < \deg a$. The elements represented by $(a(x), b(x))$ with $a(x)$ of small degree can be used for index calculus [16].

### III. Scheme

This scheme uses four basic points, the first one is to use of Diffie-Hellman process to exchange the keys then encoding a message into a sequence of point, encrypted the points and finally decrypted it as shown in Fig 1.

#### A. Diffie-Hellman Key Exchange.

Using Hyper-Elliptic Curve Diffie-Hellman in the algorithm generates the key which is far secure and efficient than the other algorithm [12]. The key is generated in HECDH as a shared key between two users. Both users A and B have a key pair as a private key and public key. User A choose a random number $K_A$ as a private key, User B also choose a private key $K_B$.

$$K_A \times P_B = K_A \times (K_B \times G) = K_B (K_A \times G) = K_B \times P_A$$

#### B. Encoding a Message into a Sequence of Point

Take a plaintext file has to be encrypted to a sequence of point. User A can encrypt the ASCII code of each and every character of the plaintext file [17].

1. Choose a Hyper-Elliptic curve.
2. Take a message for encryption.
3. The characters of message are converted to its ASCII value, m.
4. Now choose an integer $k$ (both users know the value of $k$).
5. To calculate $x = m \times k + 1$.
6. To obtain $y$ from put the value of x in the Hyper-Elliptic Curve Equation.

We get the point of each character as $P_m = (x, y)$. In this way, the message becomes a sequence of point.

#### C. Encryption

After having generated a public key, $P_B = K_B \times G$, and base point on the Hyper-Elliptic Curve To encrypt and send a message $P_m$ from User A to user B, A choose a random positive integer $k_A$ as a private key of User A and generates the cipher text $C_m$ consisting points.

$$C_m = \{ P_m + K_A P_B \}$$

Where, $P_{B+} = K_B \times G$, is the public key of user B, G is base point of HEC.

Note that user A has masked the message $P_m$ by adding $K_A \times P_B$ to it. No one but User A knows the value of $k_A$ because $K_A$ is the private key of User A, so nobody can remove the mask $K_A \times P_B$. And then User A sends the Cipher text $C_m$ as an encrypted form to User B.

**Algorithm:**

1. $P_m = (x, y)$. [$P_m$ is a sequence of point on curve]
2. Choose a base point from the Hyper-Elliptic Curve, $G = (x_1, y_1)$.
3. Then calculate

$$C_m = (P_m + K_A P_B)$$

Where,

- $C_m$= cipher text point on the curve,
- $K_A$= private key of user A,
- $P_m$= plaintext point on the curve,
- $P_B$= public key of user B.

4. $P_m = (x, y)$; $P_B = K_B \times G$.
5. $P_m + K_A \times P_B = (x, y) + K_A \times P_B$. [$K_A \times P_B$ = Shared Key as a point on curve$(x_2,y_2)$]
6. From the above we calculate the adding of two points.
7. Let the two points as $(x, y)$ and $(x_2, y_2)$.
8. Slope, $S = (y - y_2) / (x - x_2)$
9. $X_3 = S^2 - x - x_2$; $Y_3 = -y + S(x - x_3)$
10. $C_m = (x_3, y_3)$
11. Repeat step 1 to step 9 until the plaintext point will null.
12. $C_m$ is a sequence of cipher text point on the curve.
D. Decryption

C<sub>m</sub> is the Cipher text point on the curve, to decrypt the cipher text point, User B multiply the private key of User B, K<sub>B</sub> and the public key of User A, P<sub>A</sub>. And subtract the result, K<sub>B</sub> × P<sub>A</sub> from the C<sub>m</sub>, which gives

\[
P_m = \frac{K_B \times P_A}{K_B \times P_B} \times (K_B \times G)
\]

Now we get point P<sub>m</sub>. P<sub>m</sub> is the plaintext point on the Hyper-Elliptic Curve. From the coordinates of this point we find the ASCII value of the input character, m. At the final we get the character from its ASCII value, m.

**Algorithm:**

1. C<sub>m</sub> = (x<sub>3</sub>, y<sub>3</sub>)
2. We first multiply the private key of user B(K<sub>B</sub>) with the public key of user A (P<sub>A</sub>),
   
   \[ K_B \times P_A \]
3. (K<sub>B</sub> × P<sub>A</sub>) is subtract from the C<sub>m</sub> = (x<sub>3</sub>, y<sub>3</sub>).
4. Let (K<sub>B</sub> × P<sub>A</sub>) = (x<sub>2</sub>, y<sub>2</sub>) = (K<sub>A</sub> × P<sub>A</sub>) [from Diffie-Hellman Key Exchange]
5. Subtract (x<sub>2</sub>, y<sub>2</sub>) from the (x<sub>3</sub>, y<sub>3</sub>).
6. Slope,
   
   \[ S = \frac{y_3 + y_2}{x_2 - x_3} \]
7. \[ x_4 = (S^2 - x_2 - x_3); \quad y_4 = y_2 - S(x_2 - x_4) \]
8. We get P<sub>m</sub> = (x<sub>4</sub>, y<sub>4</sub>) = (x, y).
9. P<sub>m</sub> is a plaintext point on the curve.
10. From x coordinate of point P<sub>m</sub>, and calculate m.
    
    \[ m = x/k \]

Finally, we get a character of message from its ASCII value, m.

**Figure 1: Flow chart of the algorithm**
IV. EXPERIMENTAL RESULT

Table 1. Comparison Table

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Security.bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>80</td>
</tr>
<tr>
<td>ECC</td>
<td>160</td>
</tr>
<tr>
<td>HECC</td>
<td>60</td>
</tr>
</tbody>
</table>

ECC and HECC are two public key algorithms. We can comparison these two cryptographic algorithms in key size as same security level as shown in Table 1. HECC required a small parameter than ECC at same security level. In which smaller parameter will makes computing faster, shorter key and the computation speed of HECC is faster than the ECC. For example, ECC key size is 160 bit and HECC key size is 60 bit at the 80 bit of security level, their key size ratio is 1 to 3. When the security level is raised to 256 and their key size ratio is 1 to 3. It means that at any security level the ratio of their key size is always 1 to 3.

HECC algorithm can achieve the same encryption results of the ECC algorithm only by use shorter key, this indicates that HECC’s safety performance is better. In addition, HECC occupy a small storage space. These benefits make HECC widespread used.

B. Performance Evaluation

In this paper, the encryption and decryption algorithm uses HECC instead of ECC. In comparison with ECC, the results of the encryption and decryption times are shown in Tables 2 and 3 respectively, which indicate that encryption and decryption time of HECC are much less than those of ECC at the same security level, i.e. shown in the Table 1

<table>
<thead>
<tr>
<th>Method of Encryption</th>
<th>Key Size (bit)</th>
<th>ECC</th>
<th>HECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.05</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>0.20</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>0.29</td>
<td>0.28</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>0.38</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>0.42</td>
<td>0.40</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: A comparison for encryption time (unit: µs)

<table>
<thead>
<tr>
<th>Method of Encryption</th>
<th>Key Size (bit)</th>
<th>ECC</th>
<th>HECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.10</td>
<td>0.12</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>0.50</td>
<td>0.54</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>0.80</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td>160</td>
<td>1.10</td>
<td>1.05</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>1.15</td>
<td>1.08</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: A comparison for decryption time (unit: µs)

V. CONCLUSIONS

The first result is that HECC algorithm is faster than ECC algorithm and it uses smaller key size length at the same security level. One of the most important problems is the limitation of storage.
With the HECC algorithm, the size of key length is small and hence there is no problem in space. On the other hand, it is faster than the other public key cryptosystem such as ECC and RSA because of shorter key length of HECC is needed than the other cryptosystem. The proposed scheme reduced 40% computation cost and increase 2.5 times faster compared to existing encryption with ECC algorithm due to small key size less computation cost and increase bandwidth.

REFERENCES