Elliptic Curve Cryptography for Ciphering Images

Nikita Gupta1, Vikas Kundu2, Neha Kurra3, Shivani Sharma4, Bhagyashree Pal5
Department of Computer Engineering
Army Institute of Technology
Pune, India

Abstract— the growing dire need for more and more secure systems has led researchers worldwide to discover and implement newer ways of encryption. Public key cryptography techniques are gaining worldwide popularity for their ease and better strength. With the rapid developments of the communication and applications of multimedia techniques in recent years lead the researchers to focuses on the security of digital data over the internet. In this paper we have discussed the use of Elliptical Curve Cryptography for ciphering color images. ECC has been proved to score over RSA on the basis of its strength and speed. In this paper we have used NIST Curves for ciphering color image.

Keywords—Encryption, Elliptic Curve Cryptography

I. INTRODUCTION

With the wireless communication, online transaction on the rise, the security of these systems is one of the major concerns facing the computing world [5]. As the channel is open air, it leaves these systems completely vulnerable to unauthenticated accessibility. Some systems which include ATM machines and smart cards which can be used at varied locations demand stringent security at the cost of low processing.

The systems so far had been using the traditional RSA techniques. But where this poses a question is the increased bit lengths in the recent times and with growing number of transactions every minute, it also questions the processing time as well. Here is where the elliptic curves have come to the rescue.

Elliptical Curve Cryptography was introduced by Neal Koblitz and Victor S. Miller. It makes use of the structure of elliptic curves over finite fields. ECC has the advantage of smaller key size over the earlier public key cryptosystems in turn demanding lesser storage requirements [3].

Theoretically the strength of elliptical curves can be attributed to the ease of finding a resultant point on the curve by multiplying a given point by a random number but deciphering the number even after knowing the given and resultant point is a herculean task [5].

II. LITERATURE REVIEW

ECC has a hard problem of elliptic curve discrete logarithm. Although even after a long period of research, there is still not even a single algorithm exists which can solve this problem efficiently? That means solving elliptic curve discrete logarithms is harder than factoring for numbers of the same size. Computationally intensive hard problem leads to a stronger cryptographic system, which means that elliptic curve cryptosystems are harder to break than RSA and Diffie-Hellman. Although initially these algorithms were not used much but now these algorithms are gaining popularity. ECC has a very wide variety of applications: the US government uses it to protect internal communications, the Tor project uses it to help assure anonymity, this mechanism was used by bitcoins to prove ownership, it provides signatures in Apple's iMessage service, it is the technique which is used to encrypt DNS information with DNSCurve, and it is also used for authentication for secure Web browsing over SSL/TLS.[7]

In 1997 Robshaw and Yin analyzed performance of RSA and ECC digital signature in a RSA laboratory. They compared RSA digital signature with a key size of 1024 bit and ECDSA with 160 bit length key. They compared both algorithms according to storage requirements and computational speed. They found ECC signature generation is seven times faster and signature verification is around six times slower than RSA signature [1].

Table 1 gives the key sizes recommended by the National Institute of Standards and Technology to protect keys used in conventional encryption algorithms like the (DES) and (AES) together with the key sizes for RSA, Diffie-Hellman and elliptic curves that are needed to provide equivalent security [8].

<table>
<thead>
<tr>
<th>Symmetric Key Size</th>
<th>RSA and Diffi-Hellman Key Size</th>
<th>ECC Key Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>1024</td>
<td>160</td>
</tr>
<tr>
<td>112</td>
<td>2048</td>
<td>224</td>
</tr>
<tr>
<td>128</td>
<td>3072</td>
<td>256</td>
</tr>
<tr>
<td>192</td>
<td>7680</td>
<td>384</td>
</tr>
<tr>
<td>256</td>
<td>15360</td>
<td>521</td>
</tr>
</tbody>
</table>

III. ELLIPTIC CURVE CRYPTOGRAPHY

The major attraction by which ECC scores over other public key cryptosystems is the same level of security it provides with smaller bit lengths thereby requiring lesser storage and processing time. Informally, an elliptic curve is a kind of cubic curve but whose solutions are confined to a
The general equation of an elliptic curve looks like:
\[ y^2 = x^3 + ax + b \] [10]

A prime number \( p \) and a generator point \( a \) are given along with the equation of the curve. The operation exploited for key selection in elliptic curve cryptography comes from considering elliptic curve as an abelian group with points as elements.

Point addition: It states that to add two points \( P \) and \( Q \) we draw a line \( PQ \) through them and find the third point of intersection \( -R \) of that line and reflect it over the axis of symmetry of the curve. The resultant point \( R \) will give the addition of the two points.

Example: An elliptic curve in \( F_p \), it is a group of points which fulfill the "curve equation". The equation is:
\[ y^2 = x^3 + ax + b \text{ mod } p \] [10]

Here \( y, x, a \) and \( b \) belongs to curve \( F_p \) and they are integers modulo \( p \). The coefficients \( a \) and \( b \) are the characteristic coefficients of the curve.

1. Curve cryptosystem parameters

For converting all these mathematical basics into a cryptosystem, some parameters are defined which are sufficient for meaningful operations. There are 6 distinct values for the \( F_p \) case and they comprise the so-called "domain parameters":

a. \( p \): The prime number which defines the field in which the curve operates, \( F_p \). All point operations are taken modulo \( p \).
b. \( a, b \): The two coefficients which define the curve. These are integers.
c. \( G \): The generator (base) point. And used to determine the "start" of the curve. This is either given in point form \( G \) or as two separate integers \( g_x \) and \( g_y \).
d. \( n \): The order of the curve generator point \( G \) (for getting the number of points on the curve).
   For digital signing using ECDSA the operations are congruent modulo \( n \), not \( p \).
e. \( h \): The cofactor of the curve. It is the quotient of the number of curve-points, or \( #E(F_p) \), divided by \( n \).

2. Generating a keypair

Generating a keypair for sender and receiver.

To calculate the private key sender \( A \), choose a random integer \( n_A \), such that
\[ 0 < n_A < n_0 \]

To calculate public key of sender \( Pb_A \) use scalar point multiplication of the private key with the generator point \( G \):
\[ Pb_A = n_A.G \]

Similarly, receiver \( B \) will also calculate private key \( n_B \) and public key \( Pb_B = n_B.G \)

3. Encryption

To encrypt a message \( M \), which is encoded as a point on the elliptic curve. Randomly select \( n_A \) (Range 1 to \( n-1 \)). Compute \( C_1 = n_AG \) and \( C_2 = M + n_APb_B \) where \( Pb_B \) is the public key of receiver, \( n \) is the number of points generated on curve and \( (C_1,C_2) \) are encrypted points.

4. Decryption

To decrypt the ciphertext, \( (C_1,C_2) \), receiver multiplies the first point in the pair by \( B \)'s secret key \( n_B \) and subtracts the result from the second point:
\[ M + n_A Pb_B - n_B(n_AG) = M + n_A (n_BG) - n_B(n_AG) = M \]

IV. APPROACH

The complete process followed by us can be summarized by the following flowchart:
1. Operations performed by the sender:
   a. The image to be encrypted is obtained.
   b. Represent the image to be encrypted using OpenCV as a Mat Container.
   c. A random random number is generated using number generator n.
   d. The image matrix is divided into n parts.
   e. Every element of every matrix is divided by another random number which is stored and so are the quotient and remainder.
   f. Koblitz encoding [3] is used to generate points on the curve.
   g. The points are encrypted.
   h. The encrypted image is then sent across the network.

2. Operations performed by the receiver:
   a. The receiver receives the encrypted image.
   b. Decryption is applied to obtain the points on the curve.
   c. Decoding is done to obtain the matrix elements.
   d. The matrix elements are multiplied by the number obtained while encryption.
   e. All the matrices are merged and converted to the original image.

V. EXPERIMENTAL SETUP

Here we used OpenCV with java and C++ to perform image Encryption and Description using ECC.

Setup1:
   - OpenCV in Java for image processing operations
   - Java for ECC Encryption and Decryption

Setup2:
   - OpenCV in C++ for image processing operations
   - Java for ECC Encryption and Decryption

The analyses of both the Setups are given in Table 2 and Table 3 respectively.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P – 192</td>
<td>0.577</td>
<td>10.161</td>
<td>2.444 0.407</td>
</tr>
<tr>
<td>P – 224</td>
<td>0.617</td>
<td>3.343</td>
<td>0.430 0.301</td>
</tr>
<tr>
<td>P – 256</td>
<td>0.575</td>
<td>2.951</td>
<td>0.448 0.361</td>
</tr>
<tr>
<td>P – 384</td>
<td>0.772</td>
<td>5.264</td>
<td>2.445 0.335</td>
</tr>
<tr>
<td>P – 521</td>
<td>1.210</td>
<td>29.736</td>
<td>0.506 0.310</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P – 192</td>
<td>0.936</td>
<td>3.675</td>
<td>0.072 0.089</td>
</tr>
<tr>
<td>P – 224</td>
<td>0.999</td>
<td>3.993</td>
<td>0.084 0.086</td>
</tr>
<tr>
<td>P – 256</td>
<td>1.087</td>
<td>4.580</td>
<td>0.097 0.120</td>
</tr>
<tr>
<td>P – 384</td>
<td>1.474</td>
<td>3.914</td>
<td>0.079 0.102</td>
</tr>
<tr>
<td>P – 521</td>
<td>1.951</td>
<td>12.622</td>
<td>0.081 0.091</td>
</tr>
</tbody>
</table>

VI. RESULT

The results can be showcased by the following images at various stages of the process:

Figure 2 Original Image
VII. FUTURE SCOPE

Elliptic Curve Cryptography is such an excellent choice for doing asymmetric cryptography in portable, necessarily constrained devices right now, mainly because of the level of security offered for smaller key sizes. A popular, recommended RSA key size for most applications is 2,048 bits. For equivalent security using Elliptic Curve Cryptography, you need a key size of 224 bits. The difference becomes more and more pronounced as security levels increase (and, as a corollary, as hardware gets faster, and the recommended key sizes must be increased). A 384-bit Elliptic Curve Cryptography key matches a 7680-bit RSA key for security. There is a scope of concealing the encrypted image within another image. Because the encrypted image will attract attention and arouse interest and someone can try to decrypt it but once it’s hidden nobody will know that there is another image behind this image. In this paper, all the pixels of the image are encrypted and if the image size is very large then proposed system will takes more processing time. So, instead of encrypting all the pixels of the image we can encrypt only the important part of the image which will help in reducing the processing time.

References

[1]. Maryam Savari and Yeoh Eng Thiam, “Comparison of ECC and RSA in Multipurpose Smart Card Application”.
[2]. Elsayed Mohammed and A.E Emrarah and Kh.El-Shenawwey, “Elliptic Curve Cryptosystems on Smart Cards”.
[6]. Kamlesh Gupta1, Sanjay Silakari, “ECC over RSA for Asymmetric Encryption: A Review”
[7]. http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/2/
[9]. Santoshi Ketan Pote, Usha Mittal “Elliptic Curve Cryptographic Algorithm”
[10]. Christof Paar, Jan Pelzl ,"Understanding Cryptography"