Implementation of Elliptical Curve Cryptography

Randhir Kumar[1], Akash Anil[2]

**[1]Gyan Ganga Institute of Technology & Management,
Bhopal, Madhya Pradesh-462021 India**

**[2]Gyan Ganga Institute of Technology & Management,
Bhopal, Madhya Pradesh-462021 India**

Abstract

This paper involves the development of the Elliptical Curve Cryptography (ECC) for file formats like audio, video and Image. It is also used for the compression of same file formats. The tools are provided for hiding and retrieving data with security constraints**.**

Keywords: *ECC, DES Algorithm, Compression Technique.*

1. Introduction

Cryptography means protecting private information against unauthorized access in that situation where it is difficult to provide physical security [1] [2].

The basic idea behind the cryptography is that "If it is not possible to prevent copying of information, it is better to prevent compression." The fundamental mathematical idea to public-key cryptography is that of a hard problem and from such problems, mechanisms for public-key key exchange might be constructed. If an additional technical requirement (a trapdoor) can be designed then the hard problem might possibly be used to construct a public-key encryption or a digital signature algorithm.

While the 20-year history of public-key cryptography has seen a diverse range of proposals for candidate hard problems only two have truly stood the test of time. These problems are known as the discrete logarithm problem over a finite field and integer factorization.

2. Elliptical Curve Cryptography (ECC)

Public-key cryptography is based on the intractability of certain mathematical problems. Early public-key systems, such as the RSA algorithm, are secure assuming that it is difficult to factor a large integer composed of two or more large prime factors. For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly-known base point is infeasible [3]. The size of the elliptic curve determines the difficulty of the problem. It is believed that the same level of security afforded by an RSA-based system with a large modulus can be achieved with a much smaller elliptic curve group. Using a small group reduces storage and transmission requirements.

For current cryptographic purposes, an *elliptic curve* is a plane curve which consists of the points satisfying the equation (1)

$$y^2 = x^3 + ax + b \qquad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

along with a distinguished point at infinity, denoted "∞". (The coordinates here are to be chosen from a fixed finite field of characteristic not equal to 2 or 3, or the curve equation will be somewhat more complicated)[4]. This set together with the group operation of the elliptic group theory form an Abelian group, with the point at infinity as identity element. The structure of the group is inherited from the divisor group of the underlying algebraic variety.

3. Application of Elliptical Curves In Key Exchange

3.1 Elliptic Curve Cryptography (ECC) domain parameters

The public key cryptographic systems involves arithmetic operations on Elliptic curve over finite fields which is determined by elliptic curve domain parameters.

The ECC domain parameters over F_q is defined by the septuple as given below **D = (*q, FR, a, b, G, n, h*)**, where

- ***q***: prime power, that is q = p or q = 2^m, where p is a prime
- ***FR***: field representation of the method used for representing field elements $\in F_q$
- ***a, b***: field elements, they specify the equation of the elliptic curve E over F_q, $y^2 = x^3 + ax + b$
- ***G***: A base point represented by G= (x_g, y_g) on E (F_q)
- ***n***: Order of point G , that is n is the smallest positive integer such that nG = ***O***
- ***h***: cofactor, and is equal to the ratio $\#E(F_q)/n$, where $\#E(F_q)$ is the curve order

The primary security in ECC is the parameter n; therefore the length of ECC key is the bit length of n. For comparative length, the security of ECC keys is much more than that of other cryptosystems. That is for equivalent security, the key length of ECC key is much lesser than other cryptosystems.

3.2 Elliptic Curve protocols

Generally in the process of encryption and decryption, we have 2 entities, the one at the encryption side and the other at the decryption side. Let us assume that Alice is the person who is encrypting and Bob is the person decrypting.

3.3 Key generation

Alice's (or Bob's) public and private keys are associated with a particular set of elliptic key domain parameters (q, FR, a, b, G, n, h).
Alice generates the public and private keys as follows

1. Select a random number d, d $\in$ [1, n – 1]
2. Compare Q = dG.
3. Alice's public key is Q and private key is d.

It should be noted that the public key generated needs to be validated to ensure that it s atisfies the arithmetic requirement of elliptic curve public key. A public key Q = (x_q, y_q) associated with the domain parameters (q, FR, a, b, G, n, h) is validated using the following procedure

1. Check that Q $\neq$ ***O***
2. Check that x_q and y_q are properly represented elements of F_q
3. Check if Q lies on the elliptic curve defined by a and b.
4. Check that nQ = ***O***

4. BLOCK DIAGRAM

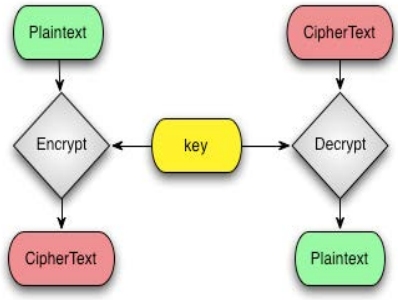

Fig.1: Structure of cryptographic system

From Fig-1 any plaintext can be hidden inside curve, image and other sources. For this purpose there is a need of private key to encrypt the plaintext to CipherText. Similarly for decrypting CipherText to Plaintext enter the same key which was used for encrypting the text.

5. Development Of Algorithm

Step1:- Select master file from embed message button
Step2:- Select any picture from the local drive
Step3:- After selecting master file select output file to embed message
Step4:- if the file should be compressed then click on check box compress
Step5:- if the message should be encrypted then Click on checkbox encrypt message
Step6:- if the message should be hidden then type message in message box and click on go button, then dialog will be appear with operation is successful or not
Step7:- close embedding message window by clicking on close button
Step8:- To retrieving encrypted, hidden, compressed message click on retrieve message button and select the output file
Step9:- click on go button and enter the encrypted password for retrieving message

6. Java Program

6.1 Program for encrypting message

```
// Encrypt the message if required

if(features== UEF || features== CEF)
{
Cipher cipher= Cipher.getInstance("DES");
SecretKeySpec
spec=new SecretKeySpec(password.substring(0, 8).getBytes(),
"DES");
cipher.init(Cipher.ENCRYPT_MODE, spec);
fileArray= cipher.doFinal(fileArray);
messageSize= fileArray.length;
}
```

In the given Program code UEF stands for Uncompressed Encrypted File and CEF stands for Compressed Encrypted File,if one of the statement is true the it will check the key which is going to store in spec object.and E ncrypt the message by using "DES" Algorithm. After encrypting the message the PlainText becoming CipherText.

6.2 Program for decrypting message

```
//Decrypt the file if required

if(features== CEF || features== UEF)
{
password= password.substring(0, 8);
byte passwordBytes[]= password.getBytes();
cipher= Cipher.getInstance("DES");
spec= new SecretKeySpec(passwordBytes, "DES");
cipher.init(Cipher.DECRYPT_MODE, spec);
try
{
fileArray= cipher.doFinal(fileArray);
                                    }
catch(Exception bp)



{
 message= "Incorrent Password";
 bp.printStackTrace();
return false;
                                    }
messageSize= fileArray.length;
                                }
```

In the given Program code UEF stands for Uncompressed Encrypted File and CEF stands for Compressed Encrypted File,if one of the statement is true the it will check the key,if entered key is wrong then give error "Incorrect Password" , if it correct then it convert the all byte value in PlaintText from CipherText.

6.3 Program for Compressing message

```
// Compress the message if required

if(features== CUF || features== CEF)
{
ByteArrayOutputStreamarrayOutputStream=new
ByteArrayOutputStream();
ZipOutputStreamzOut=new
ZipOutputStream(arrayOutputStream);
ZipEntry entry= new ZipEntry(dataFile.getName());
zOut.setLevel(compression);
zOut.putNextEntry(entry);
zOut.write(fileArray, 0, messageSize);
zOut.closeEntry();
zOut.finish();
zOut.close();
// Get the compressed message byte array
fileArray= arrayOutputStream.toByteArray();
compressionRatio= (short) ((double)fileArray.length /
(double)messageSize * 100.0);
messageSize= fileArray.length;          }
```

In the given Program code CUF stands for Compressed Unencrypted File and CEF Stands for Compressed Encrypted File, For compressing any file and data there is Class known as ZipOutputStream which stores all bytes value of file and use to make Compress.

6.4 Program for uncompressing message

```
// Uncompress the file if required
if(features== CUF || features== CEF)
{
ByteArrayOutputStream by= new ByteArrayOutputStream();
DataOutputStream out= new DataOutputStream(by);
ZipInputStream zipIn= new ZipInputStream(new
ByteArrayInputStream(fileArray));
ZipEntry entry= zipIn.getNextEntry();
dataFile= new File(entry.getName());
byteArrayIn= new byte[1024];
while((tempInt= zipIn.read(byteArrayIn, 0, 1024))!= -1)
out.write(byteArrayIn, 0, tempInt);
zipIn.close();
out.close();
fileArray= by.toByteArray();
messageSize= fileArray.length;
                    }
```

In the given Program code CUF stands for Compressed Unencrypted File and CEF Stands for Compressed Encrypted File, For decompressing any file and data there is Class known as ZipIntputStream which read 1024 byte value of file and use to make decompress.

7. Flowchart

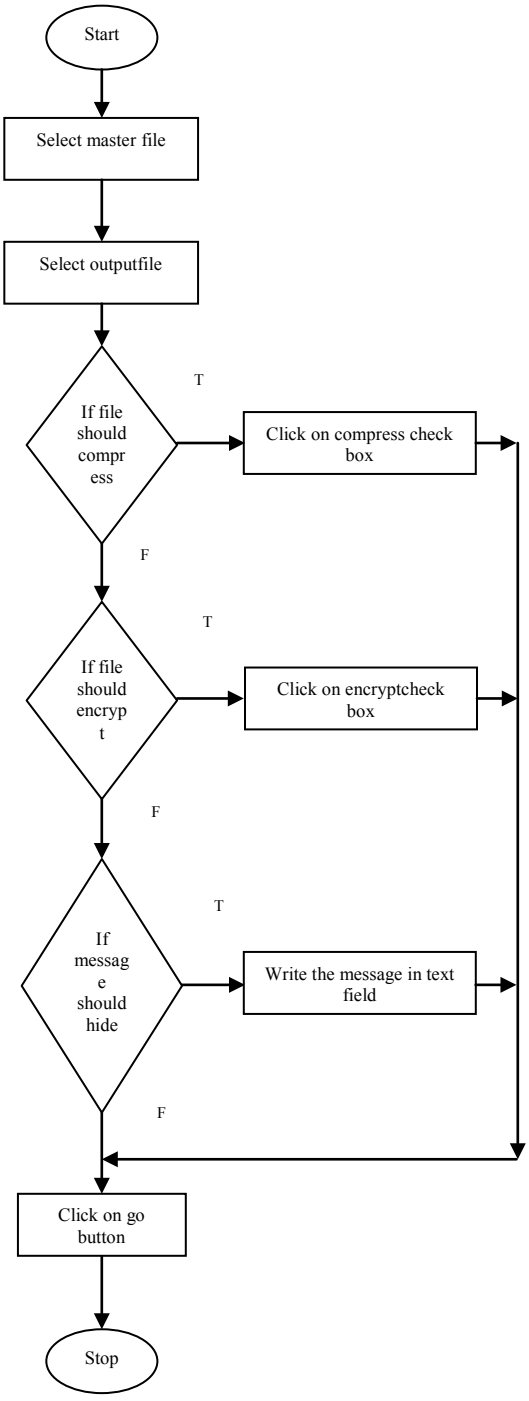

Fig.2: Flow Chart of work flow of ECC

The work flow of Elliptical Curve Cryptography is shown in Fig:-2.Three major work with ECC is Compress, Encrypt, and Hide the message. User can perform all three works simultaneously or individually. For encryption of data user has to enter private key and that key will accessed by “DES” Algorithm to make any PlainText to CipherText.

At the same time User can retrieve the CipherText to PlainText by entering the same key.

8. Results And Discussions

Fig.3: Main window of ECC

Fig.4: Encryption & Decryption Window of ECC

The main window of ECC after running the program in Fig:-3. There is four buttons namely embed message, retrieve message, embed file, retrieve file. Embed message button is use to hide the message with some source image with master file and output file which is shown in Fig-4.

In Fig-4 there is three options shown that is Compress, Encrypt, Message, while Encrypting the message there is a text field which is provided for entering the private key. And for entering the message there is text area provided.

For compression there is slider to compress the output file. After all three operation user has to choose go button to complete the task.

9. Conclusion

The data is sent not only by image but it is also sent by audio, video and image. In this paper the implementation of audio, video and image file formats are presented. The master files as well as output file has been developed. The image, audio and video can be of any format.

Acknowledgments

The authors wish to thank the management of Gyan Ganga Institute of Tech & Mgmt, Bhopal for their constant encouragement for completion of this work. The authors wish to acknowledge Mr.D.SureshBabu for his co-operation for completing his paper, also to Mr. S. M. Raj Bharath for his encouragement during this work.

References

[1] Cryptography and Elliptic Curves, http://www.tcs.hut.fi/~helger/crypto/link/public/elliptic/

[2] "Wiliam Stalling" Cryptography and network security". New Riders,June 3,2010.

[3] V. Miller, "Uses of elliptic curves in cryptography", Advances in Cryptology,Springer-Verlog New York,1986.

[4] Jeffrey L. Vagle, "A Gentle Introduction to Elliptic Curve Cryptography",BBN Technology,Nov21,2010.

.

Randhir Kumar was born in 1984 in Bihar, India. He completed his graduation in Computer Application from MCRP University, Bhopal & post graduation in Computer Application from Vellore Institute of Technology, Vellore, India. He was Engineer at Honeywell Technology Solution Lab India till 2009. Currently he is Sr.Lecturer in Department of Information Technology, Gyan Ganga Institute of Technology & management, Bhopal (M.P), India. His major research areas are data mining, web mining and Image Processing.

Akash Anil was born in 1985 in Bihar, India. He completed his B.Tech from BPUT University, Orrisa, India. He is currently with Gyan Ganga Institute of Technology & management, Bhopal (M.P), India. His major research areas are Image Processing, digital image processing and network security.