CRY TOOL

AN IMAGE ENCRYPTION USING ECC AND HILL CIPHER

TABLE OF CONTENTS

1. INTRODUCTION

1.1. OBJECTIVE

The general objective of CRY Tool is to explore Hill Cipher for image encryption, which is one among the many possible encryption schemes available for images and evaluate the security offered by the Elliptical Curve Cryptosystem based secret self-invertible 4×4 key matrix, for different input images and the same Elliptical Curve Equation. Moreover, the use of self-invertible matrix is to focus on reducing the computational cost of Hill Cipher incurred due to matrix inverse computation, without compromising of the security offered by Hill Cipher.

1.2. MOTIVATION

With the increasing use of multimedia/social media platforms for sharing images over the internet, it is important to establish secure transmission of media (here images) through the world wide web without a third-party intervention or intrusion. The ease in availability of technology and internet, has led to the striking rise in the number of internet users who share their personal images, users who share images with highly confidential information embedded in it (Stenography). Images are used in communication in a range varying from social media platforms to Geographical Information System (GIS), hence making it one of the most vulnerable to attacks without proper security.

1.3. PROBLEM STATEMENT

Encryption of a grey scale image using Hill Cipher by overcoming the overhead in computing the inverse and secure transmission of common key matrix with the assistance of an Elliptical curve Cryptosystem(ECC) to generate a self-invertible key matrix for encryption and decryption.

1.4. PROPOSED SOLUTION

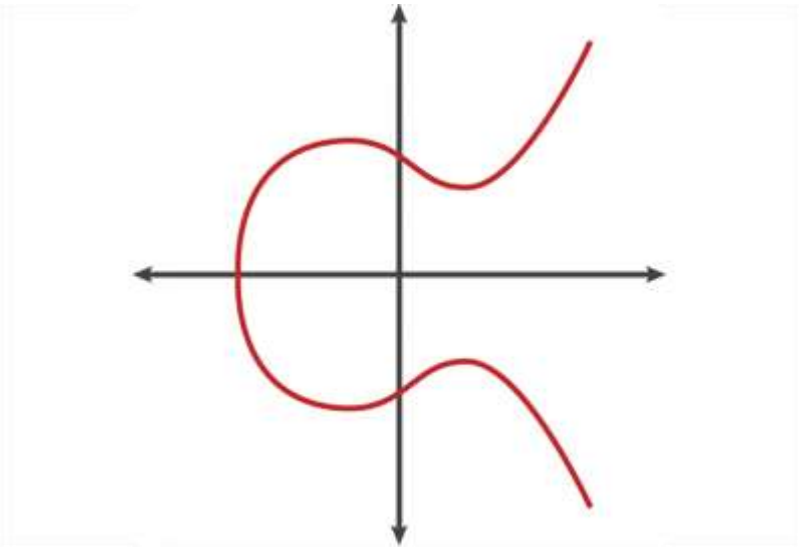
The proposed solutions encrypts/decrypts a image in a processes including two steps- Key Generation using ECC, Encryption/Decryption Process

1.4.1. Key Generation using ECC

Elliptic curve E defined over a prime field F_p is defined by

$$E: y^2 = x^3 + ax + b \pmod p$$

Where $a,b \in F_p$ and satisfy the condition $4a^3 + 27b^2 \neq 0 \pmod p$



(Fig 1.4.1.1)- An Elliptic Curve

If $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ are two points in the elliptic curves then some of the Elliptic curve operations are

:

Point Addition: Addition of the two points P_1 and P_2 gives a new point R

$$R = P_1 + P_2 = (x_3, y_3)$$

$$s = (y_2 - y_1)/(x_2 - x_1)$$

$$x_3 = (s^2 - x_1 - x_2) \pmod p$$

$$y_3 = (sx_1 - sx_3 - y_1) \pmod p$$

Point Doubling: Addition of the same point P1 gives a new point R

$$R=2P_1=(x_3,y_3)$$

$$s=(3x_1^2+a)/(2y_1)$$

$$x_3=(s^2-2x_1)(\text{mod } p)$$

$$y_3=(sx_1-sx_3-y_1)(\text{mod } p)$$

Scalar Multiplication: Scalar multiplication of a point P by an integer k is defined by repeated addition of the point P with itself, and resultant point R also lies of the elliptical curve.

Elliptic Curve cryptography uses a process similar to that of Diffie Hellman Key exchange algorithm but instead of discreate logarithms the fact that given the initial generator point(A generator point is a point that generates a cyclic group within a given elliptic curve.) and final point after scalar multiplication of an elliptical curve generator point with scalar k, it is computationally difficult find the value of k is used as the one-way function.

A 2×2 key matrix is generated using a series of scalar multiplication defined over a point in the elliptic curve which forms the basis for creating the other elements of the 4×4 self-invertible matrix (discussed in detail in Design chapter)

1.4.2. Encryption

The image is divided into blocks of length four and is multiplied with the 4×4 self-invertible matrix to obtain the corresponding encrypted pixel values/ciphered pixels using which the ciphered image can be generated. This ciphered image is transmitted through the insecure channel.

1.4.3. Decryption

The Ciphered image generated from the cipher pixels is yet again divided into blocks of length four and this vector is multiplied with the common secret 4×4 self-invertible matrix to obtain the original pixel values and hence the original image.

A major drawback in the proposed solution is that the image should have column Pixel values as a multiple of 4 only, as the pixels are divided into groups of four.

2. DESIGN

In a previous section, we have discussed about the proposed solution and in this chapter, we shall see a detailed description of the solution design.

2.1. SUB KEY GENERATION USING ECC

User A (The Sender)

a) Choose The Private Key $N_a \in [1,P-1]$
b) Compute The Public Key $P_a = N_a.G$ {scalar multiplication}
c) Compute The Initial Key $K_I = N_a.P_b = (x,y)$ {scalar multiplication}
d) Compute $K_1 = x.G = (k_{11},k_{12})$
   And $K_2 = y.G = (k_{21},k_{22})$
e) Generate the Self-Invertible Key Matrix K_m

User B (The Receiver)

a) Choose The Private Key $N_b \in [1,P-1]$
b) Compute The Public Key $P_b = N_b.G$ {scalar multiplication}
c) Compute The Initial Key $K_I = N_b.P_a = (x,y)$ {scalar multiplication}
d) Compute $K_1 = x.G = (k_{11},k_{12})$
   And $K_2 = y.G = (k_{21},k_{22})$
e) Generate the Self-Invertible Key Matrix K_m

2.2. SELF-INVERTIBLE MATRIX

The self-invertible matrix K_m, is compute by both the users from K_1 and K_2 which respectively forms the rows of the 2×2 matrix K_{11}

$$K_m = \left[\begin{array}{c|c} K_{11} & K_{12} \\ \hline K_{21} & K_{22} \end{array}\right].$$

$$K_m = \left[\begin{array}{cc:cc} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ \hdashline k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{array}\right]$$

(Fig 3.2.1)- the self-invertible matrix

Where,

$K_{12} = I - K_{11}$

$K_{21} = I + K_{11}$

$K_{22} = (-1)\, K_{11}$

I is a unit matrix of order 2×2

2.3. ENCRYPTION

Divide the image pixel values into blocks of size four P=[P1; P2; P3; . . .](where P1=P2=… are vectors of order 4×1) and each block in P, will be multiplied by the self-invertible key matrix K_m under mod 256(as gray scale image is used) to get the ciphered blocks C=[C1; C2; C3; . . .]. Create the Ciphered image from C.

$$\text{Let } P_1 = \begin{bmatrix} p_{11} \\ p_{21} \\ p_{31} \\ p_{41} \end{bmatrix} \text{ then}$$

$$C_1 = K_m . \quad P_1 = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{21} \\ p_{31} \\ p_{41} \end{bmatrix}$$

$$= \begin{bmatrix} ((k_{11}p_{11} + k_{12}p_{21} + k_{13}p_{31} + k_{14}p_{41}) \bmod 256 \\ ((k_{21}p_{11} + k_{22}p_{21} + k_{23}p_{31} + k_{24}p_{41}) \bmod 256 \\ ((k_{31}p_{11} + k_{32}p_{21} + k_{33}p_{31} + k_{34}p_{41}) \bmod 256 \\ ((k_{41}p_{11} + k_{42}p_{21} + k_{43}p_{31} + k_{44}p_{41}) \bmod 256 \end{bmatrix}$$

$$= \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \\ c_{41} \end{bmatrix}$$

(Fig 3.3.1)- Encryption Process

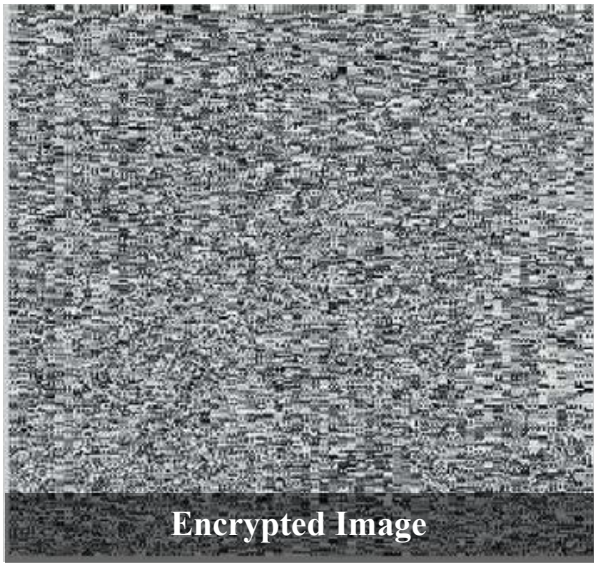
2.4. DECRYPTION

Divide the image pixel values into blocks of size four C=[C1; C2; C3; . . .](where C1=C2=… are vectors of order 4×1) and each block in P, will be multiplied by the self-invertible key matrix K_m under mod 256 to get the original blocks P=[P1; P2; P3; . . .]. Create the Original image from P.

3. TESTING AND RESULTS

3.1. EXPECTED RESULTS

(Fig 5.1.1)- expected output



3.2. OUTPUTS

Following are few of the test cases that were tried in the code that was implemented

ORIGINAL IMAGE	CIPHERED IMAGE	DECRYPTED IMAGE	Inference
			Example for a poor Encryption. Elliptic curve function should be chosen according to the input image
			Relatively better encryption with data loss
			Relatively better encryption with data loss

(Fig 5.1.1)- outputs after implementation