Image cryptography using Elliptic curve and Magic matrix with Advanced encryption standard

by

Deril Raju, BE/10489/16

Muppidi Sai Pranav, BE/10488/16

Lalitha Eleswarapu, BE/10435/16

Project Report

Submitted in partial fulfillment of requirement for the award in the degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

Supervised by

Dr. Rupesh Kumar Sinha

Assistant Professor

Department of Electronics and Communication Engineering

Department of Electronics and Communication Engineering

Birla Institute of Technology, Mesra, Ranchi – 835215

May 2019

DECLARATION CERTIFICATE

This is to certify that the work presented in this project entitled " **Image cryptography using Elliptic curve and Magic matrix with Advanced encryption standard**", in partial fulfillment of the requirement for the award of Degree of **Bachelor of Engineering in Electronics & Communication Engineering**, submitted to the Department of Electronics & Communication Engineering of Birla Institute of Technology, Mesra, Ranchi, Jharkhand is a bona fide work carried out by **Deril Raju ,Muppidi Sai Pranav and Lalitha Eleswarapu** under my supervision and guidance.

To the best of my knowledge, the content of this project, either partially or fully, has not been submitted to any other institution for the award of any other degree.

Date: 10/5/2019

Dr. R K Lal
Department of ECE
Birla Institute of Technology
Mesra, Ranchi.

Department of Electronics and Communication Engineering

Birla Institute of Technology, Mesra, Ranchi – 835215

CERTIFICATE OF APPROVAL

This is to certify that the project entitled "**Image cryptography using Elliptic curve and Magic matrix with Advanced encryption standard**" is hereby approved as a suitable design of an engineering subject, carried out and presented in satisfactory manner to warrant its acceptance as prerequisite to the degree for which it has been submitted.

It is understood that by this approval, the undersigned do not necessarily endorse any conclusion drawn or opinion expressed therein, but approve the project for which it is submitted.

Internal Examiner	External Examiner
Date:	Date:

Dr. S. Pal
Head of Department
Department of Electronics and Communication Engineering
Birla Institute of Technology
Mesra, Ranchi

ACKNOWLEDGEMENT

We owe our deepest gratitude to our advisor **Dr. Rupesh Kumar Sinha**, for his constant support and motivation, despite his extremely busy schedule. Our interactions with him always resulted in new ideas and proved beneficial towards our work. Without his constant presence and supervision our work would not have been successful.

We are very grateful to **Dr. S. Pal, Professor and Head, Department of Electronics and Communication Engineering, Birla Institute of Technology, Mesra** for extending all the facilities at all times for pursuing this course.

And lastly our batch mates who have always been there with valuable suggestions and support in our endeavors.

Deril Raju

Muppidi Sai Pranav

Lalitha Eleswarapu

ABSTRACT

Recently, many image encryption models have been proposed using elliptic curve cryptography and magic matrix. But with security being the highest priority while transferring sensitive information, models nowadays are required to withstand robust hacking without compromising on the speed of the system. This work proposes an image encryption scheme using a combination of elliptic curves, magic matrix and AES (Advanced Encryption Standard) in order to provide greater security to the image encryption operation. The pixel intensity values of the image are first transformed by a substitution method utilising elliptic curves. These pixels are subjected to another round of value transformation, before being scrambled according to the magic matrix algorithm in the second stage of the encryption process. Scrambling ensures position transformation of the pixels in the image. The last stage of encryption uses the AES technique to generate a 128-bit key and encrypt the entire image in a block-by-block manner. This was done to provide an additional layer of security, and obtain better results when compared to other encryption models. The proposed technique is simulated using MATLAB. After the process of image encryption and decryption, the performance of the model was assessed using parameters like correlation coefficient, NPCR (Number of pixels change rate) and UACI (Unified average changing intensity).

CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Internet communication is playing an important role in transferring large amounts of data in various fields. Some of the data might be transmitted through insecure channels from sender to receiver. Different techniques and methods have been used by private and public sectors to protect sensitive data from intruders because the security of electronic data is a crucial issue.

Cryptography is one of the most significant and popular techniques to secure the data from attackers by using two vital processes that are Encryption and Decryption. Encryption is the process of encoding data to prevent it from intruders from reading the original data easily. This stage has the ability to convert the original data (Plaintext) into an unreadable format known as Cipher text.

The next process that has to be carried out by the authorized person is Decryption. Decryption is contrary to encryption. It is the process to convert cipher text into plain text without missing any words in the original text. To perform these processes cryptography relies on mathematical calculations along with some substitutions and permutations with or without a key. Modern cryptography provides confidentiality, integrity, nonrepudiation and authentication. These days, there are a number of algorithms available to encrypt and decrypt sensitive data which are typically divided into three types. First one is symmetric cryptography ; the same key is used for encryption and decryption data. Second one is Asymmetric cryptographic. The proposed model has the following block diagram

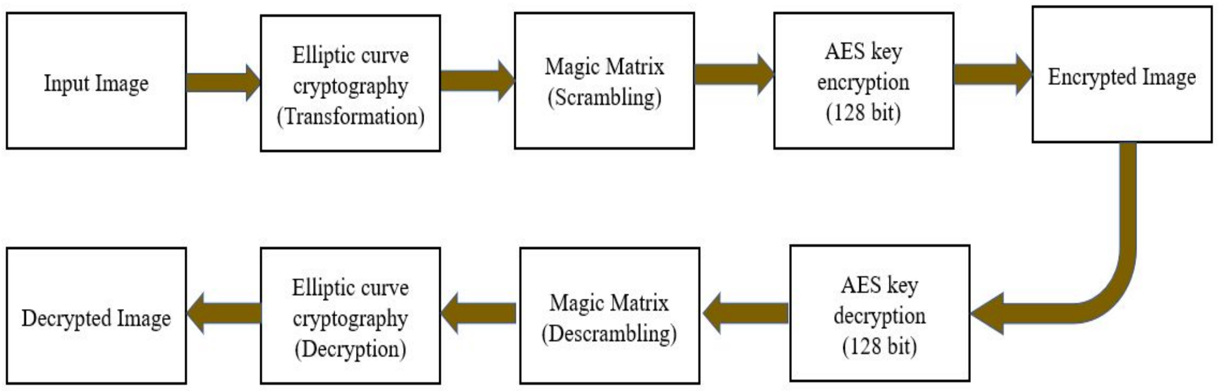

Input Image
Elliptic curve cryptography (Transformation)
Magic Matrix (Scrambling)
AES key encryption (128 bit)
Encrypted Image
AES key decryption (128 bit)
Magic Matrix (Descrambling)
Elliptic curve cryptography (Decryption)
Decrypted Image

Fig 1: Block Diagram

ELLIPTIC CURVE CRYPTOGRAPHY

1.1 Introduction

In 1985, elliptic curves were independently applied by Miller and Koblitz to introduce a new public key cryptosystem. Following this, many researchers tried to employ it on different data types; and thus improve their efficiency by proposing various techniques. In fact, ECC has attractive advantages that motivated cryptographers to use it. It provided greater security and a more computationally efficient performance, with equivalent key sizes, in comparison to other public keys. This characteristic changed ECC into an acceptable choice for real time multimedia applications. Due to large sizes and high data rates of multimedia data types, such as images, videos, and audio, a cryptosystem using a short key size, with high security, was needed

The principal attraction of elliptic curve cryptography over other techniques is that it provides equal security for a smaller bit size, thus reducing the processing overhead. A 512 bit ECC key offers the same security as a 15360 bit RSA key, due to the exponential algorithms used by ECC. This technique is ideal for constrained environments such as pager, PDAs, cellular phones and smart cards.

The properties of the elliptic curve along with the complexity of the underlying elliptic curve discrete logarithm problem (ECDLP), make it very tough to decipher. The ECDLP can be explained as; Given points P, Q $\in$ E(Fq), to find an integer a, if it exists, such that Q = kP. Given P and Q, it is computationally infeasible to obtain k, if k is sufficiently large. Hence, k is the discrete logarithm of Q to P. We can see that the main operation involved in ECC is point multiplication, namely, multiplication of a scalar k with any point P on the curve to obtain another point Q on the curve. This problem is the fundamental building block for elliptic curve cryptography and pairing-based cryptography, and has been a major area of research in computational number theory and cryptography for several decades.

1.2 The Elliptic Curve

An elliptic curve is a set of points defined by the equation,

$$y^2 \bmod p = (x^3 + a*x + b) \bmod p$$

where $4*a^3 + 27*b^2 \neq 0$, in order to exclude singular curves. ‘a’ and ‘b’ are known as domain parameters, while ‘p’ determines the range of the curve. Based on the values of ‘a’ and ‘b’, the elliptic curve assumes different shapes on the plane. Elliptic curves are also symmetric about the x - axis. Elliptic curves are used for value transformation in our model. ‘x’ represents the original pixel intensity value while ‘y’ represents the transformed value.

An example of an elliptic curve is shown below

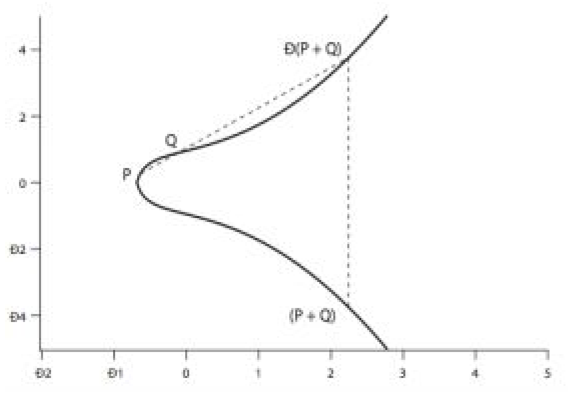


Fig 2. The elliptic curve

As seen in the figure, if we consider two points on the curve, P and Q; a line passing through these points will intersect at a third point on the curve. The inverse of this point gives us the sum of the two points (P + Q).

Multiplication of a point with a positive integer k is defined as the sum of copies of P, k times. This operation is called point multiplication in ECC.

1.3 Methodology

1.3.1 Encryption

An elliptic curve $E_p(a, b)$ was chosen. Input image was read in the form of a matrix, with pixel intensity values ranging from 0 to 255. An auxiliary base parameter 'k' was chosen (considerations must be made to select a value that is optimal, for example k=10, since large values lead to computational load on the processor, while small values may result in pixels being unencrypted).

An attempt was made to solve the curve at x = mk+1, where 'm' represents an individual pixel intensity value. If the curve could not be solved using x = mk+1, x= mk+2,3,4 up to x= mk+(k-1) were given as inputs to the elliptic curve equation; and a value of '*y*' was reached at or before x = mk + (k-1), which represents the transformed image pixel value.

The transformed values 'y' for respective pixel intensities 'x' on the elliptic curve can range from 0 to 'p'. Since pixel intensity values lie between 0 (black) and 255 (white), and diminishing the value of 'p' or base parameter 'k' could either lead to decreasing the range of operation or decreasing the probability of solving a curve for a particular value of x, the modulo operator (mod 255) was used on all outputs obtained by solving the curve.

The quotients obtained were stored in a separate array, and could later be used to reconstruct the 'y' values, which can be used for decrypting the original pixel intensity values.

1.3.2 Decryption

The elements in the encrypted image matrix and the quotient array were used to recover the mapped pixel intensity values

y(mapped) = q (quotient) * 255 + y^1 (corresponding element of encrypted image matrix)

The 'x' value corresponding to the transformed value 'y' on the curve was found using the lookup table consisting of points on the elliptic curve. The original pixel intensity value between 0 and 255 was then solved for using the formula,

p(original pixel value) = floor[(x-1)/k]

where floor(x) represents the greatest integer less than or equal to 'x'.

MAGIC MATRIX

2.1 INTRODUCTION

In combinatorial design, a magic square is a NxN square grid (where N is the number of cells on each side) filled with distinct positive integers in the range 1 to N^2 ($N>=3$) such that each cell contains a different integer and the sum of the integers in each row, column and diagonal is equal. A square grid with N cells on each side is said to have order N. The sum is called the magic constant or magic sum of the magic square. The magic constant or magic sum of the magic square is given by the formula $N(N^2+1)/2$ where N is the order of the square.

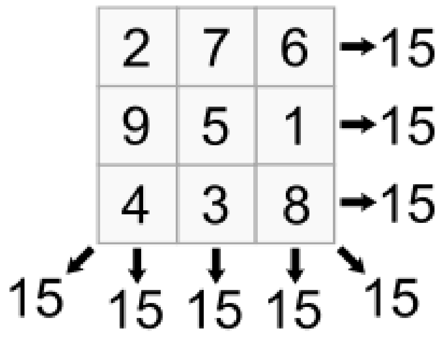


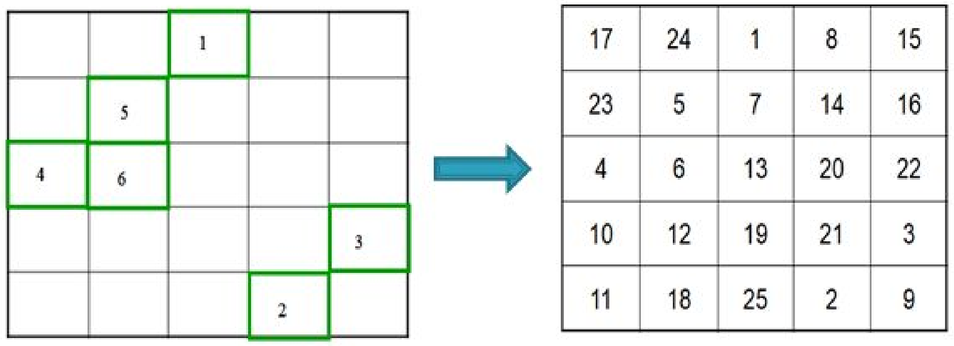
Fig 3: 3x3 magic square with magic constant 15

A weak upper bound put on the total number of magic squares possible for a given order N is given by a formula $(N^2)!/(8(2N+1)!)$. In the book 'The Zen of Magic Squares, Circles and Stars' by C. Pickover, it is stated that the total number of different magic squares of size 3 is 1, size 4 = 880 and size 5 = 275,305,224. For the 6×6 case, there are estimated to be approximately 1.8×10^{19} squares.

In the method proposed, the concept of magic matrix is used in image cryptography. The pixel values in the image matrix are encrypted using the '**magic matrix**' of the corresponding order. The encryption using the magic matrix scrambles the pixel values of the original image matrix.

Construction of magic matrix:

1. You start by placing the number 1 in any of the cells of your N x N magic square. (A '1' in the top middle will give a perfect magic square, however, you can place the '1' anywhere, the diagonals might not sum to the magic number.)
2. The next step is to place the next successive integer in the square above and to the right of the "1".
3. Continue this last step until the square is filled.
4. The numbers wrap around the square, so when you reach the top of the square, wrap to the bottom row, and if you reach the right side, then wrap to the left.
5. When you come to the upper right corner drop down one row to continue filling numbers.
6. If you go to place the next number, x, in a cell that is already filled, then place x in the cell below x-1, the number you had just placed.



17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

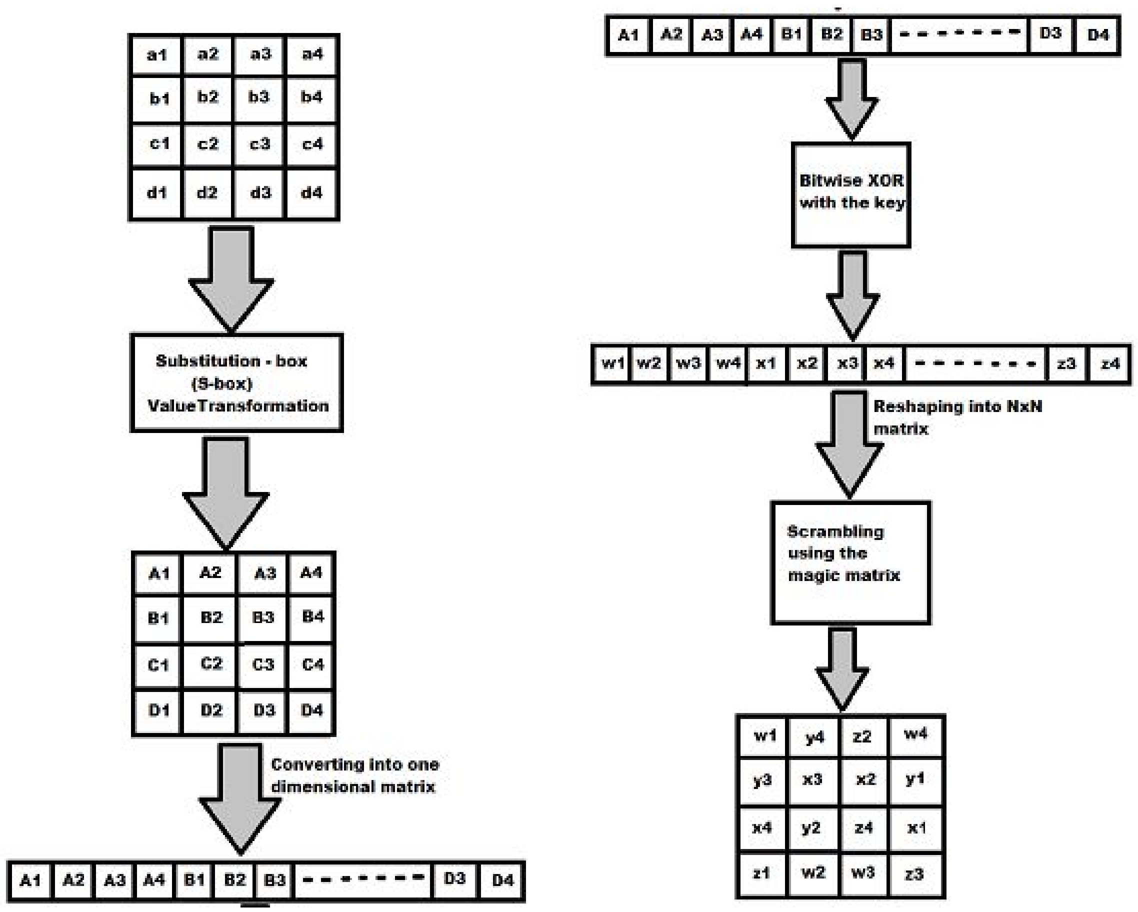
Fig 4: Construction of Magic Matrix

2.2 METHODOLOGY:

2.2.1 Encryption:

The methodology proposed in the encryption of images using magic matrix involves both **value** and **position transformation**. The method involves a 16byte key and a substitution box (S-box) for value transformation along with scrambling using the magic matrix. The two dimensional image matrix is traversed and each pixel value is transformed by looking up the S-box. The resultant two dimensional matrix is then converted into a one dimensional matrix. The one dimensional matrix is divided into chunks each with a size of 16bytes. For any remaining pixels, an extra chunk is considered. Bitwise XOR operation is performed between the 16 byte key and each chunk and the value of the pixel is replaced with this XORed value. While looping through the chunks, the key is circularly rotated by 1 byte in every loop. Hence, the key is changed in every iteration while performing XOR operation on one dimensional image array chunks. Magic matrix of order equal to the dimension of the image matrix is created. Hence it has numbers from 1 to (NxN) arranged. Now traversing through each element in the magic matrix, the value at that position is taken as an index. The pixel value located at this index of the one dimensional image array replaces the element of the magic matrix which is the index itself. In this way all the NxN elements of the magic matrix are replaced by the elements by mapping them to their positions using the indices. The output is the encrypted image.

Consider [a1,a2,a3….d3,d4] as the input image matrix.The following figure shows the step wise encryption process resulting in [w1,w2,w3....z3,z4] as the output encrypted image matrix.



a1 a2 a3 a4
b1 b2 b3 b4
c1 c2 c3 c4
d1 d2 d3 d4
Substitution - box (S-box) ValueTransformation
A1 A2 A3 A4
B1 B2 B3 B4
C1 C2 C3 C4
D1 D2 D3 D4
Converting into one dimensional matrix
A1 A2 A3 A4 B1 B2 B3 - - - - - - - D3 D4
A1 A2 A3 A4 B1 B2 B3 - - - - - - - D3 D4
Bitwise XOR with the key
w1 w2 w3 w4 x1 x2 x3 x4 - - - - - - - z3 z4
Reshaping into NxN matrix
Scrambling using the magic matrix
w1 y4 z2 w4
y3 x3 x2 y1
x4 y2 z4 x1
z1 w2 w3 z3

Part-1 Part-2

Fig 5: Encryption using Magic Matrix

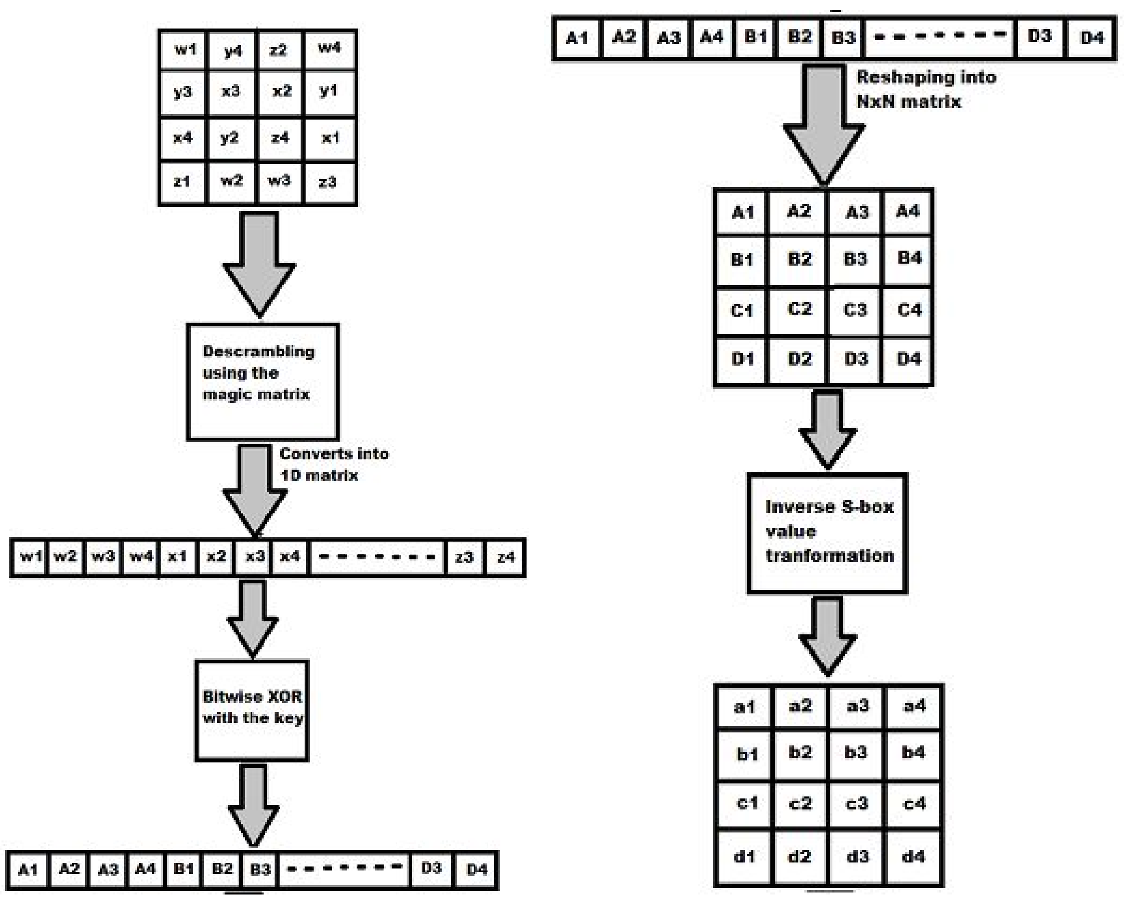
2.2.2 Decryption:

The decryption of the magic matrix involves the descrambling of the input matrix using the magic matrix, 16 byte key operation followed by the value restoration using the inverse S-box. An 1xN^2 one dimensional matrix with all zeros and a NxN magic matrix are generated. Traverse through the input matrix as well as the magic matrix simultaneously. With the element in the magic matrix as an index of the one dimensional matrix, fill that index of the one dimensional matrix with the element at the input matrix in the same position as the magic matrix.This will result in a one dimensional descrambled matrix.

The one dimensional matrix is then divided into chunks of size 16byte each. For any remaining pixels, an extra chunk is considered. The same key operation as done in the encryption end should be done on the decryption end. Bitwise XOR operation is performed between the 16 byte key and each chunk and the value of the pixel is replaced with this XORed value. While looping through the chunks, the key is circularly rotated by 1 byte in every loop. The rotation of the key should be similar to the rotation done in the encryption end.

This one dimensional matrix after performing the XOR operation is then reshaped into a two dimensional matrix of the dimension NxN. The reshaped matrix is traversed and each value is mapped back using the inverse S-box and the value is replaced by the mapped value. The resultant matrix will be the decrypted image matrix.

Consider [w1,w2,...z3,z4] as the input encrypted image matrix.The following figure shows the step wise decryption process resulting in [a1,a2,...d3,d4] as the output decrypted image matrix.


w1 y4 z2 w4
y3 x3 x2 y1
x4 y2 z4 x1
z1 w2 w3 z3
Descrambling using the magic matrix
Converts into 1D matrix
w1 w2 w3 w4 x1 x2 x3 x4 - - - - - - - z3 z4
Bitwise XOR with the key
A1 A2 A3 A4 B1 B2 B3 - - - - - - - D3 D4
A1 A2 A3 A4 B1 B2 B3 - - - - - - - D3 D4
Reshaping into NxN matrix
A1 A2 A3 A4
B1 B2 B3 B4
C1 C2 C3 C4
D1 D2 D3 D4
Inverse S-box value tranformation
a1 a2 a3 a4
b1 b2 b3 b4
c1 c2 c3 c4
d1 d2 d3 d4

Part - 1　　　　　　　　　　Part - 2

Fig 6 : Decryption using Magic magic

ADVANCED ENCRYPTION STANDARD

3.1 Introduction

Cryptography plays an important role in the security of data transmission. This paper addresses efficient hardware implementation of the AES (Advanced Encryption Standard) algorithm and describes the design and performance testing of Rijndael algorithm. A strong focus is placed on high throughput implementations, which are required to support security for current and future high bandwidth applications. This implementation will be useful in wireless security like military communication and mobile telephony where there is a gayer emphasis on the speed of communication. This standard specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128,192, and 256 bits. Throughout the remainder of this standard, the algorithm specified herein will be referred to as ―the AES algorithm. The algorithm may be used with the three different key lengths indicated above, and therefore these different ―flavors‖ may be referred to as ―AES-128‖, ―AES192‖, and ―AES-256‖.

AES is short for Advanced Encryption Standard and is a United States encryption standard defined in Federal Information Processing Standard (FIPS) 192. AES is the most recent of the four current algorithms approved for federal use in the United States. AES is a symmetric encryption algorithm processing data in blocks of 128 bits. AES is symmetric since the same key is used for encryption and the reverse transformation, decryption. The only secret necessary to keep for security is the key. AES may be configured to use different key-lengths, the standard defines 3 lengths and the resulting algorithms are named AES-128, AES-192 and AES-256 respectively to indicate the length in bits of the key. The older standard, DES or Data Encryption Standard. DES is up to 56bits only. To overcome the disadvantages of des algorithm, the new standard is AES algorithm.

3.2 AES Algorithm Specification

An implementation of the AES algorithm shall support at least one of the three key lengths: 128, 192, or 256 bits (i.e., Nk = 4, 6, or 8, respectively). Implementations may optionally support two or three key lengths, which may promote the interoperability of algorithm implementations. For the AES algorithm, the length of the Cipher Key, K, is 128, 192 or 256 bits. The key length is represented by Nk = 4, 6, or 8which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by Nr, where Nr = 10 when Nk = 4, Nr = 12 when Nk = 6, and Nr = 14 when Nk = 8. The only Key-Block-Round combinations that conform to this standard are given in the below Table.

Bit Pattern	Key Length (NK Words)	Block Size (NB Words)	No Of Rounds (NR Words)
AES 128	4	4	10
AES 192	6	4	12
AES 256	8	4	14

Table 1. Key-Block-Round Combinations

3.3 Encryption

In encryption mode, the initial key is added to the input value at the very beginning, which is called an initial round. This is followed by 9 iterations of a normal round and ends with a slightly modified final round, as one can see in Figure 2. During one normal round the following operations are performed in the following order: Sub Bytes, Shift Rows, Mix Columns, and Add Round key. The final round is a normal round without the Mix Columns stage.

Steps in AES

- Sub Bytes - a non-linear substitution step where each byte is replaced with another according to a lookup table.
- Shift Rows - a transposition step where each row of the state is shifted cyclically a certain number of steps
- Mix Columns - a mixing operation which operates on the columns of the state, combining the four bytes in each column .
- Add Round Key - each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule

3.3.1 SubBytes Transformation

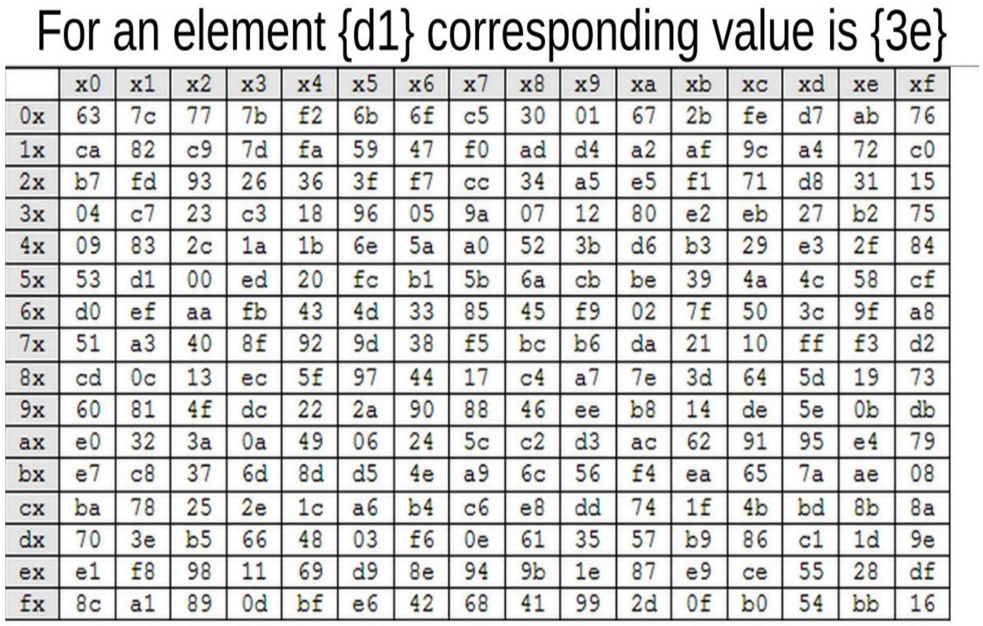
The Sub Bytes transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box which is invertible is constructed by composing two transformations:

1. Take the multiplicative inverse in the finite field GF (28), the element {00} is mapped to itself.

2. Apply the following affine transformation (over GF (2)): For 0<i<8, where bi is the ith bit of the byte, and ci is the ith bit of a byte c with the Value {63} or {01100011}. Here and elsewhere, a prime on a variable Indicates that the variable is to be updated with the value on the right. In matrix form, the affine transformation element of the S-box can be expressed as:

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Fig 7: Affine Transformation



For an element {d1} corresponding value is {3e}

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig 8: S-Box

3.3.2 Shift Rows Transformation

In the Shift Rows transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row is not shifted at all, the second row is shifted by one the third row by two, and the fourth row by three bytes to the left. Specifically, the Shift Rows transformation proceeds as follows:

The shift value shift(r,Nb) depends on the row number, r, as follows (recall that Nb = 4): shift(1,4) 1; shift(2,4) 2 ; shift(3,4) 3 .This has the effect of moving bytes to ―lower‖ positions in the row (i.e., lower values of c in a given row), while the ―lowest‖ bytes wrap around into the ―top‖ of the row (i.e., higher values of c in a given row)

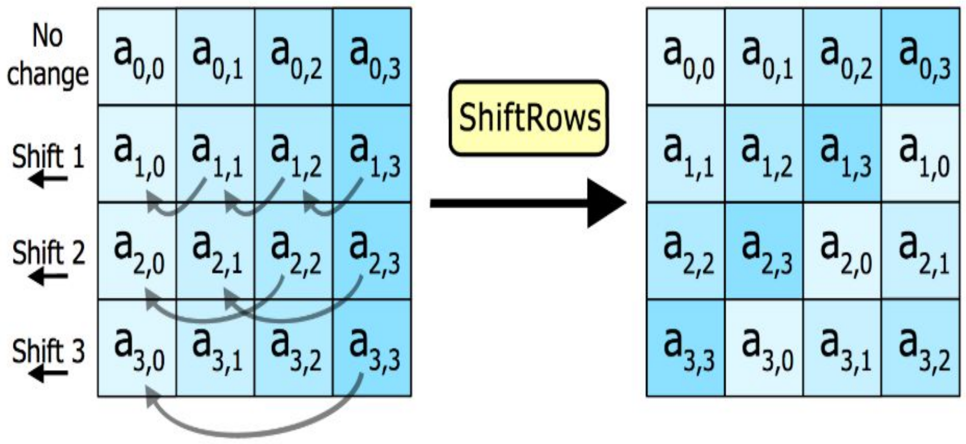

No change
Shift 1
Shift 2
Shift 3
ShiftRows
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,1}$ $a_{1,2}$ $a_{1,3}$ $a_{1,0}$
$a_{2,2}$ $a_{2,3}$ $a_{2,0}$ $a_{2,1}$
$a_{3,3}$ $a_{3,0}$ $a_{3,1}$ $a_{3,2}$

Fig 9: Shift Rows Transformation

3.3.3 Mix Columns Transformation

The Mix Columns transformation operates on the State column-by-column, treating each column as a four-term polynomial.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$(0 \le c < N_b)$$

Fig 10: Mix columns polynomial

As a result of this multiplication, the four bytes in a column are replaced by the following:

- S’0,c = ({02} • s0,c) + ({03} • s1,c) + s2,c + s3,c
- S’1,c = s0,c + ({02} • s1,c) + ({03} • s2,c) + s3,c
- S’2,c = s0,c + s1,c + ({02} • s2,c) + ({03} • s3,c)
- S’3,c = ({03} • s0,c) + s1,c + s2,c + ({02} • s3,c)

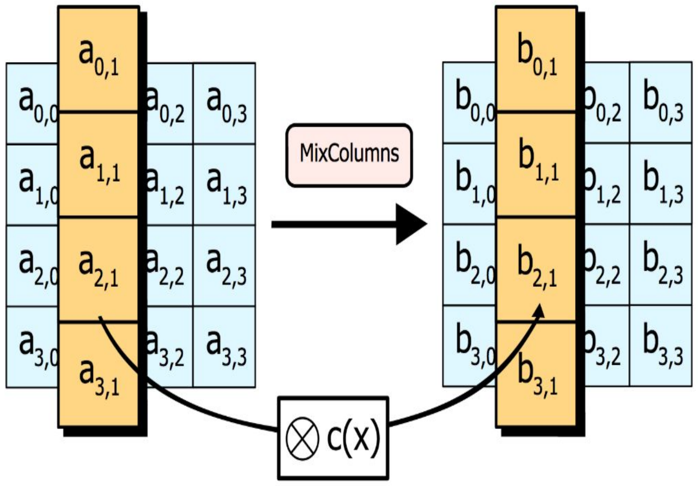

$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
MixColumns
$\otimes c(x)$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

Fig 11: Mix Columns Transformation

3.3.4 Add round Key Transformation

In the Add Round Key transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of Nb words from the key schedule.Those Nb words are each added into the columns of the State, such that [wi] are the key schedule words, and round is a value in the range 0 round Nr. In the Cipher, the initial Round Key addition occurs when round = 0, prior to the first application of the round function. The application of the Add Round Key transformation to the Nr rounds of the Cipher occurs when 1<round <Nr.

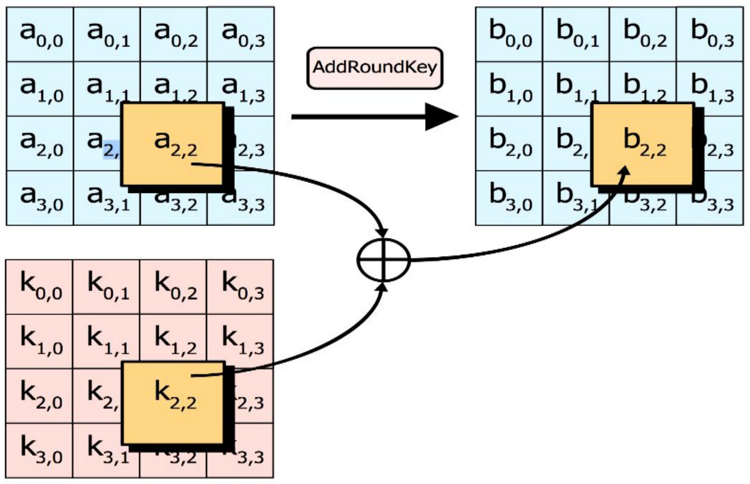
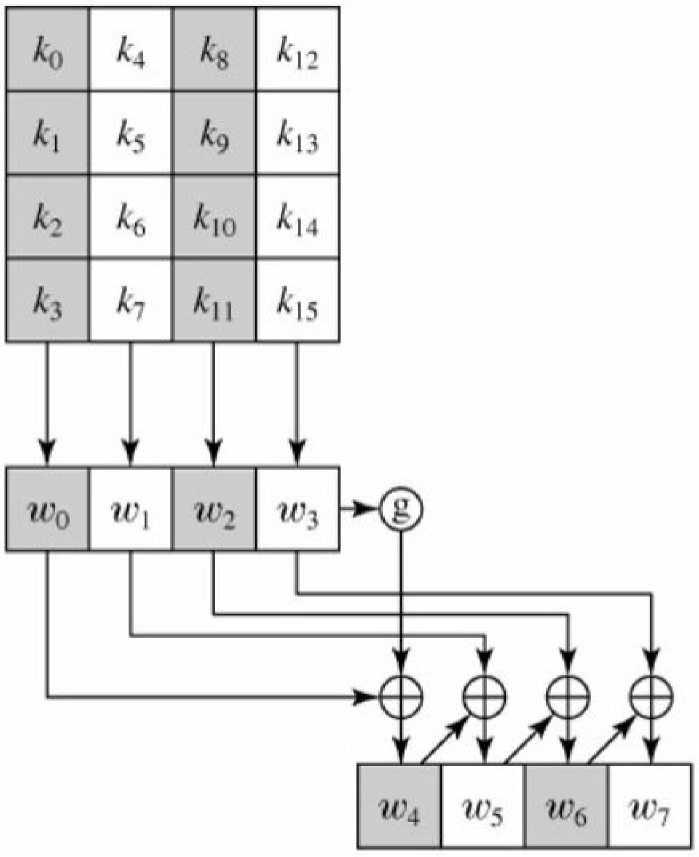


Fig 12: Add round key

3.4 Key Expansion in AES

The AES algorithm takes the Cipher Key, K, and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb (Nr + 1) words: the algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [wi], with i in the range 0 < i < Nb(Nr + 1). The expansion of the input key into the key schedule proceeds according to the pseudo code.

SubWord is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function Rot Word takes a word [a0,a1,a2,a3] as input, performs a cyclic permutation, and returns the word [a1,a2,a3,a0]. The round constant word array, Rcon[i], contains the values given by [xi-1,{00},{00},{00}], with x i-1 being powers of x (x is denoted as {02}) in the field GF(28). The first Nk words of the expanded key are filled with the Cipher Key. Every following word, w[i], is equal to the XOR of the previous word, w[i-1], and the word Nk positions earlier, w[i-Nk]. For words in positions that are a multiple of Nk, a transformation is applied to w[i-1] prior to the XOR, followed by an XOR with a round constant, Rcon[i]. This transformation consists of a cyclic shift of the bytes in a word (RotWord), followed by the application of a table lookup to all four bytes of the word (SubWord). It is important to note that the Key Expansion routine for 256-bit Cipher Keys (Nk = 8) is slightly different than for 128- and 192-bit Cipher Keys. If Nk = 8 and i-4 is a multiple of Nk, then SubWord () is applied to w [i-1] prior to the XOR.



k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7

Fig 13: AES Key expansion

3.5 Decryption

In decryption mode, the operations are in reverse order compared to their order in encryption mode. Thus it starts with an initial round, followed by 9 iterations of an inverse normal round and ends with an AddRoundKey. An inverse normal round consists of the following operations in this order: AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes. An initial round is an inverse normal round without the InvMixColumns.

3.5.1 Inverse Shift rows Transformation

InvShiftRows is the inverse of the ShiftRows transformation. The bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, r = 0, is not shifted. The bottom three rows are cyclically shifted by Nb - shift(r, Nb) bytes, where the shift value shift(r,Nb) depends on the row number.

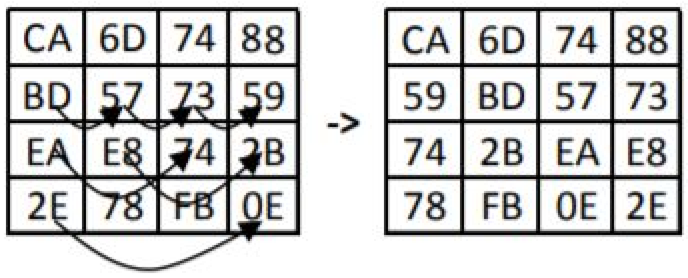
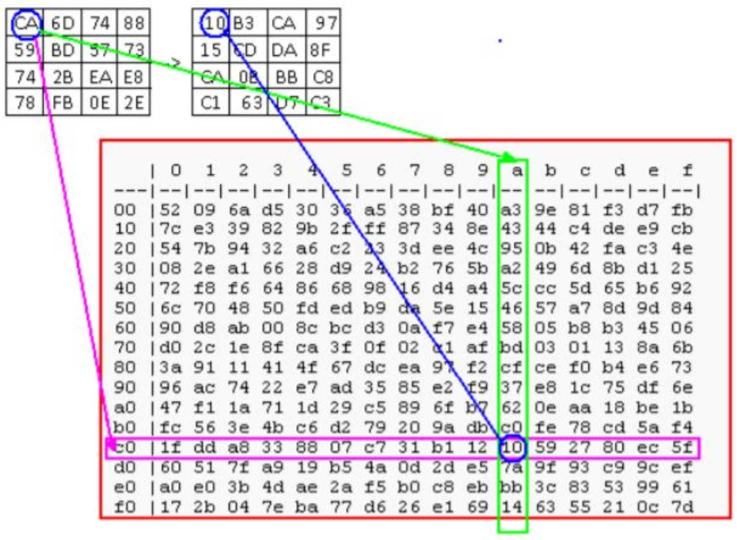


Fig 14: Inv shift rows transformation

3.5.2 Inverse SubBytes Transformation

InvSubBytes is the inverse of the byte substitution transformation, in which the inverse Sbox is applied to each byte of the State. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF (28).The inverse S-box used in the InvSubBytes () transformation


CA 6D 74 88
59 BD 57 73
74 2B EA E8
78 FB 0E 2E
->
10 B3 CA 97
15 CD DA 8F
CA 0B BB C8
C1 63 D7 C3

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
70	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Fig 15: Inv SubBytes Transformation

3.5.3 Inverse MixColumns Transformation

InvMixColumns is the inverse of the MixColumns transformation. InvMixColumns operates on the State column-by-column, treating each column as a four term polynomial. The columns are considered as polynomials over GF (28) and multiplied modulo x4 + 1 with a fixed polynomial a-1(x), given by a-1(x) = {0b} x3 + {0d} x2 + {09} x + {0e}, this can be written as a matrix multiplication. Let As a result of this multiplication, the four bytes in a column are replaced by the following:

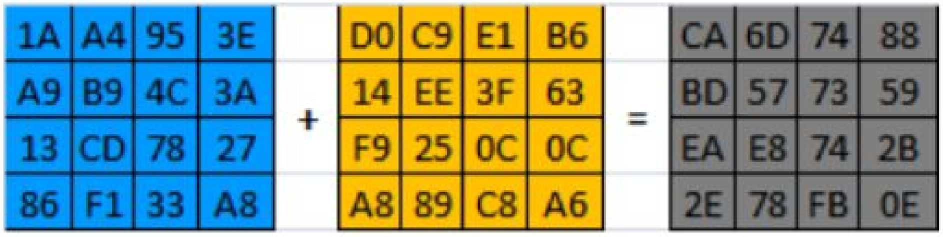
$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Fig 16: Inv MixColumns

- S’0,c = ({0e} • s0,c) + ({0b} • s1,c) + ({0d} • s2,c) + ({09} • s3,c)
- S’1,c = ({09} • s0,c) + ({0e} • s1,c) + ({0b} • s2,c) + ({0d} • s3,c)
- S’2,c = ({0d} • s0,c) + ({09} • s1,c) + ({0e} • s2,c) + ({0b} • s3,c)
- S’3,c = ({0b} • s0,c) + ({0d} • s1,c) + ({09} • s2,c) + ({0e} • s3,c)

3.5.4 Inverse Add round Key

AddRoundKey is its own inverse, since it only involves an application of the XOR operation. Equivalent Inverse Cipher transformations differ from that of the Cipher, while the form of the key schedules for encryption and decryption remains the same. However, several properties of the AES algorithm allow for an Equivalent Inverse Cipher that has the same sequence of transformations as the Cipher (with the transformations replaced by their inverses). This is accomplished with a change in the key schedule. The two properties that allow for this Equivalent Inverse Cipher are as follows: The Sub Bytes and Shift Rows transformations commute; that is, a Sub Bytes transformation immediately followed by a Shift Rows transformation is equivalent to a Shift Rows transformation immediately followed by a Sub Bytes transformation. The same is true for their inverses, InvSubBytes and InvShiftRows. The column mixing operations - MixColumns and InvMixColumns – are linear with respect to the column input, which means Inv MixColumns(state XOR Round Key) =InvMixColumns(state)XORInvMixColumns(RoundKey).The se properties allow the order of InvSubBytes and InvShiftRows transformations to be reversed. The order of the AddRoundKey and InvMixColumns transformations can also be reversed, provided that the columns (words) of the decryption key schedule are modified using the InvMixColumns transformation. The equivalent inverse cipher is defined by reversing the order of the InvSubBytes and InvShiftRows transformations and by reversing the order of the AddRoundKey and InvMixColumns transformations used in the —round loop‖ after first modifying the decryption key schedule for round = 1 to Nr-1 using the InvMixColumns transformation. The first and last Nb words of the decryption key schedule shall not be modified in this manner.



1A	A4	95	3E		D0	C9	E1	B6		CA	6D	74	88
A9	B9	4C	3A	+	14	EE	3F	63	=	BD	57	73	59
13	CD	78	27		F9	25	0C	0C		EA	E8	74	2B
86	F1	33	A8		A8	89	C8	A6		2E	78	FB	0E

Fig 17: Inv Add round key

Results and Analysis

The following parameters are used to evaluate the performance of an encryption scheme.

NPCR & UACI: NPCR refers to Number of changing pixel rate and UACI refers to Unified Averaged Changed Intensity. The NPCR measures the percentage of different pixels that change in two images which is the encrypted image when only one pixel is changed in the original image, while UACI measures the average intensity of differences between the original and encrypted images. —The number of changing pixel rate (NPCR) and the unified averaged changed intensity (UACI) are two most common quantities used to evaluate the strength of image encryption algorithms/ciphers with respect to differential attacks. Conventionally, a high NPCR/UACI score is usually interpreted as a high resistance to differential attacks. However, it is not clear how high NPCR/UACI is such that the image cipher indeed has a high security level. Further, these theoretical values are used to form statistical hypothesis NPCR and UACI tests. Critical values of tests are consequently derived and calculated both symbolically and numerically. As a result, the question of whether a given NPCR/UACI score is sufficiently high such that it is not discernible from ideally encrypted images is answered by comparing actual NPCR/UACI scores with corresponding critical values. Experimental results using the NPCR and UACI randomness tests show that many existing image encryption methods are actually not as good as they are purported, although some methods do pass these randomness tests.

Entropy: Entropy is a basic and important concept in information theory introduced by Claude E. Shannon. It is also often used as a measure of the unpredictability of a cryptographic key in cryptography research areas. Ubi-comp has emerged rapidly as an exciting new paradigm. Together with these trends, applied cryptography and security have become rising big issues for providing secure and trusted computing in the next generation of information and communications. A detailed discussion of these issues includesapplied cryptography and security concerns that cover amongst others, confidentiality, integrity,and availability including various application areas. In particular, these topics will comprehensively focus on the important aspects of entropy-based applied cryptography and enhanced security for Ubi-comp. Topics in Ubi-comp include entropy-based applied cryptographic aspects, entropy-based hash functions, mathematical and algorithmic foundations of applied cryptography.

$$E(I) = \sum_{i=0}^{n} P(I_i) \log_2 P(I_i)$$

Correlation Coefficient: Correlation test consists of randomly selecting N pairs of adjacent pixels (vertical, horizontal, and diagonal) from the original and the encrypted images separately. Then, the correlation coefficient of each pair is calculated using

$$\begin{aligned} e(x) &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ d(x) &= \frac{1}{N}\sum_{i=1}^{N}(x_i - e(x))^2 \\ \operatorname{cov}(x,y) &= \frac{1}{N}\sum_{i=1}^{N}(x_i - e(x))(y_i - e(y)) \\ r_{xy} &= \frac{\operatorname{cov}(x,y)}{\sqrt{d(x)}\sqrt{d(y)}} \end{aligned}$$

Key Sensitivity: An ideal image encryption procedure should be sensitive to the secret key. It means that a little bit change in a secret key should produce completely different image. Encryption algorithms should also have a high sensitivity to encryption keys. This means that any small change in the key should lead to a significant change in the encrypted, or decrypted, image.

4.1 Comparison of Magic Matrix (MM) and Advanced Encryption Standard (AES)

The below is a comparison for the blocks Magic matrix and Advanced Encryption Standard (AES). This analysis is done to compare the Proposed block and the actual encryption standard used worldwide. The test is done on two standard and two test images, so that we can have a clear idea about the performance on both standard and real life images.

Standard Image I : Lena

Blocks	**Parameters**				
	Correlation Coefficient		**Entropy**	**Sensitivity Analysis**	
	With Encrypted Image	With Decrypted Image		**NPCR**	**UACI**
Magic Matrix	0.0151	0.9987	7.966	0.9958	0.2840
AES	0.0069	0.9998	7.9885	0.9957	0.2826

Table 7: MM vs AES Lena image

Standard Image II : Camera Man

Blocks	**Parameters**				
	Correlation Coefficient		**Entropy**	**Sensitivity Analysis**	
	With Encrypted Image	With Decrypted Image		**NPCR**	**UACI**
Magic Matrix	-0.0182	0.9987	7.9039	0.9974	0.331
AES	-0.0168	0.9998	7.9814	0.9956	0.332

Table 8: MM vs AES Cameraman Image

Test Image I : Girl

Blocks	**Parameters**				
	Correlation Coefficient		**Entropy**	**Sensitivity Analysis**	
	With Encrypted Image	With Decrypted Image		**NPCR**	**UACI**
Magic Matrix	0.0025	0.9977	7.9805	0.9957	0.304
AES	-0.0019	0.9998	7.9888	0.9976	0.305

Table 9: MM vs AES Girl Image

Test Image II : BIT Mesra

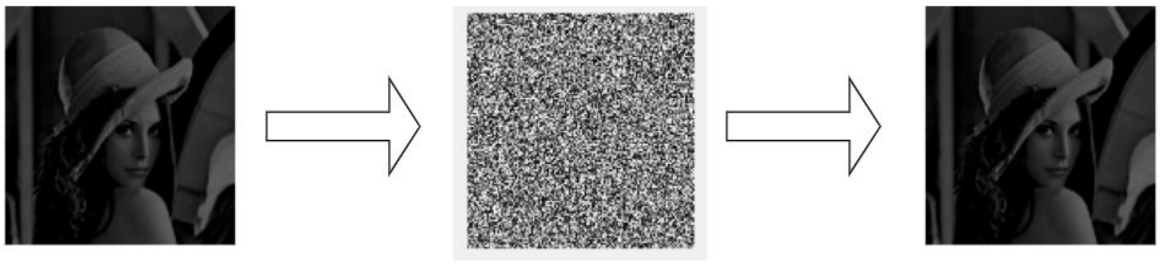
Blocks	**Parameters**				
	Correlation Coefficient		**Entropy**	**Sensitivity Analysis**	
	With Encrypted Image	With Decrypted Image		**NPCR**	**UACI**
Magic Matrix	-0.00021	0.9997	7.9793	0.9964	0.320
AES	0.00979	0.9998	7.9870	0.9963	0.321

Table 10: MM vs AES BIT Image

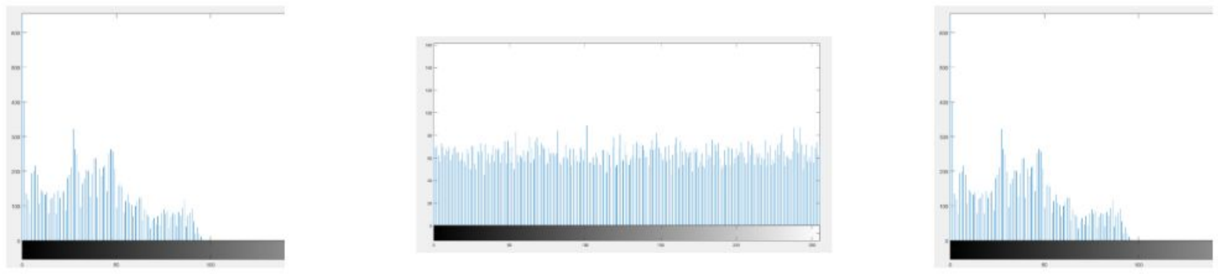
‘

4.2 The below 3 test results are for standard images used for image cryptography.

Results: Lena Image



Input Image: Lena Encrypted Image Decrypted Image



Histogram: Lena image Histogram: Encrypted image Histogram: Decrypted

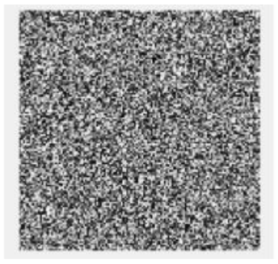
Fig 18: Lena image encryption and decryption

Methods	Parameters				
	Correlation Coefficient		**Entropy**	**Sensitivity Analysis**	
	With Encrypted Image	With Decrypted Image		**NPCR**	**UACI**
Proposed Method (lena image)	-0.000658	0.9887	7.998	0.9959	0.3068
Enhanced 3D logistic map[13]	−0.0237	0.9658	7.991	0.9940	0.335
Double spiral scan and chaotic map[12]	0.0032	0.9757	7.997	0.9972	0.335

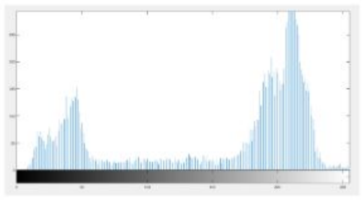
Table 2: Lena image value comparison

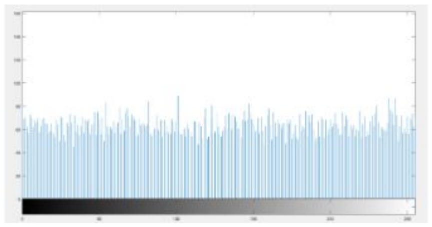
Results: Cameraman Image

Input Image: Cameraman


Encrypted Image

Decrypted Image


Histogram: Cameraman image


Histogram: Encrypted image

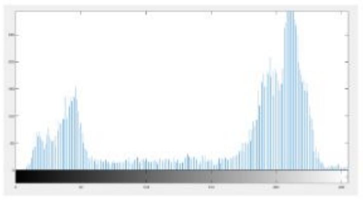
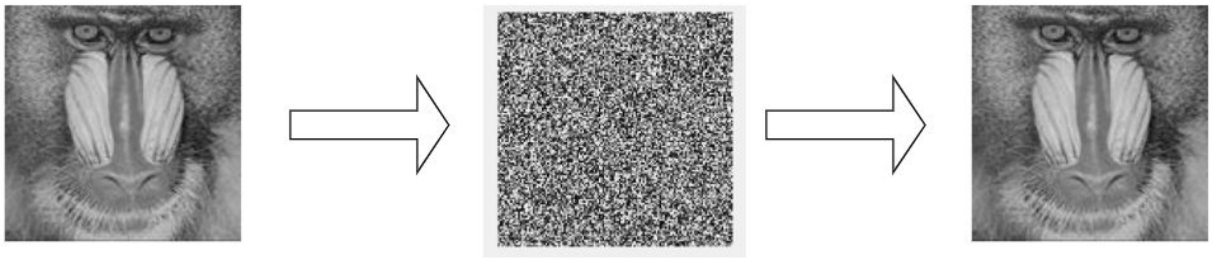

Histogram:Decrypted

Fig 19: Cameraman image encryption and decryption

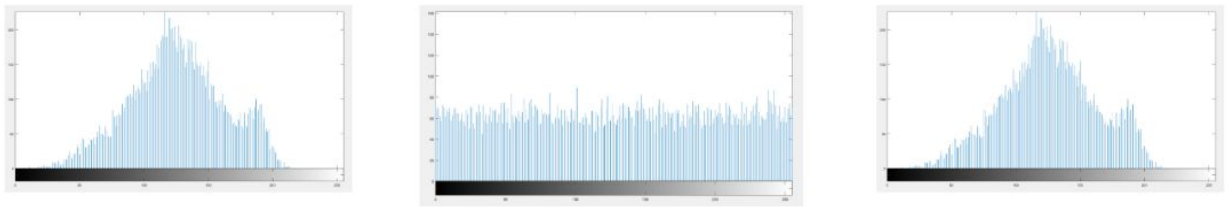
Methods	**Parameters**				
	Correlation Coefficient		**Entropy**	**Sensitivity Analysis**	
	With Encrypted Image	With Decrypted Image		**NPCR**	**UACI**
Proposed Method (cam man image)	0.00261	0.9887	7.991	0.9956	0.3268
Enhanced 3D logistic map[13]	−0.0537	0.9858	7.989	0.9840	0.331
Double spiral scan and chaotic map[12]	-0.0009	0.9796	7.997	0.9962	0.334

Table 3: Cameraman image value comparison

Results: Baboon Image



Input Image: Baboon Encrypted Image Decrypted Image



Histogram: Baboon image Histogram: Encrypted image Histogram:Decrypted

Fig 20: Baboon image encryption and decryption

Methods	**Parameters**				
	Correlation Coefficient		**Entropy**	**Sensitivity Analysis**	
	With Encrypted Image	With Decrypted Image		**NPCR**	**UACI**
Proposed Method (baboon image)	-0.0027	0.9887	7.998	0.9958	0.3168
Enhanced 3D logistic map[13]	−0.137	0.9758	7.991	0.9960	0.335
Double spiral scan and chaotic map[12]	-0.0073	0.9576	7.997	0.9972	0.334

Table 4: Baboon image value comparison

4.3 The below 2 test results are for Random images for testing the model

Results: Girl Image

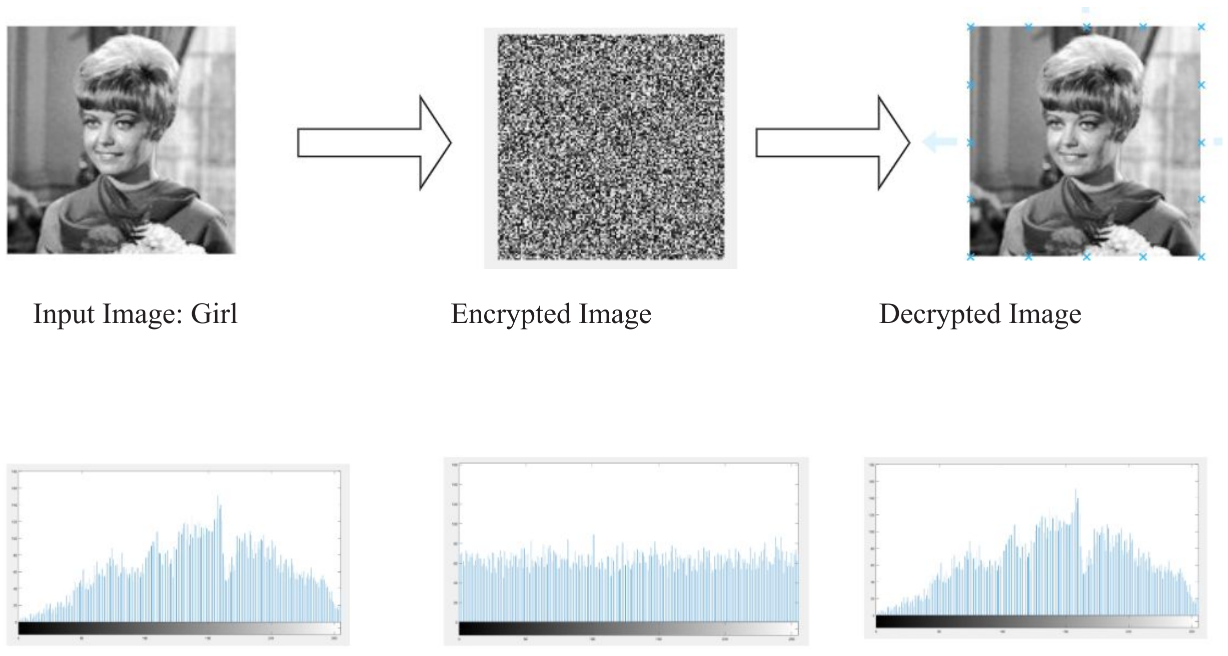


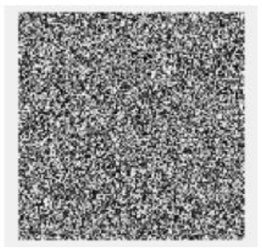
Fig 21: Girl image encryption and decryption

Methods	Parameters				
	Correlation Coefficient		Entropy	Sensitivity Analysis	
	With Encrypted Image	With Decrypted Image		NPCR	UACI
Proposed Method (girl image)	0.0013	0.9887	7.988	0.9962	0.3056

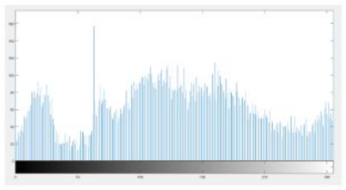
Table 5: Girl image value comparison

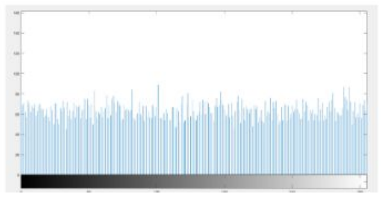
Results: BIT Mesra Image

Input Image: BIT Mesra


Encrypted Image

Decrypted Image


Histogram: BIT image


Histogram: Encrypted image

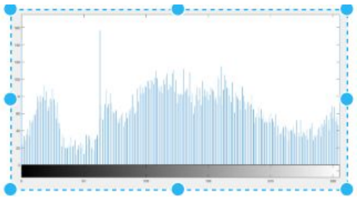

Histogram: Decrypted

Fig 22: BIT Mesra image encryption and decryption

Methods	Parameters				
	Correlation Coefficient		Entropy	Sensitivity Analysis	
	With Encrypted Image	With Decrypted Image		NPCR	UACI
Proposed Method (BIT image)	-0.0068	0.9887	7.990	0.9957	0.3219

Table 6: BIT Mesra image value comparison

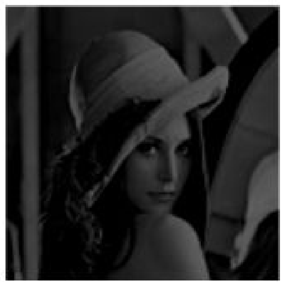
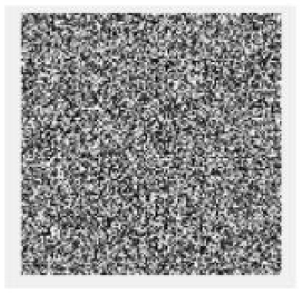
4.4 Key Sensitivity Analysis

The below results shown are for decrypted outputs when the key is changed slightly (One nibble in this case)

Correct Key: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

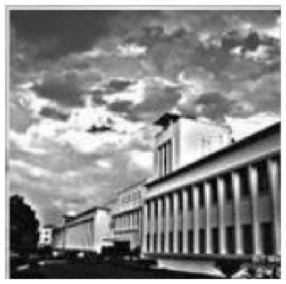
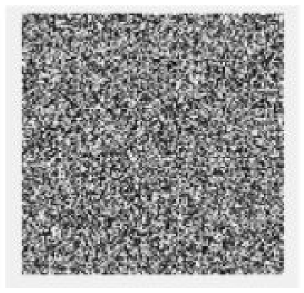
Wrong Key: 00 01 02 04 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F (One Nibble change)

Standard Image (Lena)

Original Image Decrypted (Correct key) Decrypted (Wrong key)

Test Image (BIT)

Original Image Decrypted (Correct key) Decrypted (Wrong key)

Fig 23: Key sensitivity analysis

CONCLUSION

In this work, we have proposed a model for image encryption that offers greater security and efficient scrambling of image pixels. We have used a combination of elliptic curve cryptography, magic matrix scrambling and advanced encryption standard (AES) in this technique, and security analysis shows that the proposed encryption scheme has almost ideal entropy and correlation coefficients, i.e. there is minimum correlation between the original and encrypted image and an almost exact recovery of the original image after decryption. Upon analysis using parameters such as NPCR and UACI, it produces better results when compared to other encryption models. We also have compared our proposed block with the widely used encryption scheme, AES.

FUTURE WORK

1. To expand the scope of the proposed idea to RGB images (True coloured images).
2. Implement Steganography for a robust and secure key transmission.
3. To explore the properties of magic matrix and increase the strength of the proposed method by using a unique way of filling the magic matrix for a given order out of many possible matrices.

REFERENCES

[1] Padma Bh, D.Chandravathi, P.Prapoorna Roja, “Encoding And Decoding of a Message in the implementation of Elliptic Curve Cryptography using Koblitz’s Method” International Journal on Computer Science and Engineering, Vol. 02, No. 05, 2010, 1904-1907.

[2] F. Amounas, E.H. El Kinani,”Security Enhancement of Image Encryption Based on Matrix Approach using Elliptic Curve”, International Journal of Engineering Inventions, e-ISSN: 2278-7461, p-ISSN: 2319-6491 Volume 3, Issue 11 (June 2014) PP: 8-16.

[3] Srinivasan Nagaraj, Dr.G.S.V.P.RAJU, K.Koteswara rao, “Image Encryption Using Elliptic Curve Cryptography and Matrix”, International Conference on Intelligent Computing, Communication & Convergence, (ICCC-2014), doi: 10.1016/j.procs.2015.04.182.

[4] Megha Kolhekar and Anita Jadhav ”implementation of elliptic curve cryptography on text and image”, International Journal of Enterprise Computing and Business Systems, ISSN (Online): 2230-8849, 2011, vol. 1, Issue 2.

[5] Laiphrakpam Dolendro Singh, Khumanthem Manglem Singh, “Image Encryption using Elliptic Curve Cryptography”, Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015), doi: 10.1016/j.procs.2015.06.054

[6] Priya Ramasamy, Vidhyapriya Ranganathan, Seifedine Kadry, “An Image Encryption Scheme Based on Block Scrambling, Modified Zigzag Transformation and Key Generation Using Enhanced Logistic—Tent Map” 21, 656; doi:10.3390/e21070656.

[7] Md. Billal Hossain, Md.Toufikur Rahman, A B M Saadmaan Rahman, “A new approach of image encryption using 3D chaotic map to enhance security of multimedia component”, Conference: 2014 International Conference on Informatics, Electronics & Vision (ICIEV) DOI: 10.1109/ICIEV.2014.6850856.

[8] Samiksha Sharma, Vinay Chopra, “Analysis Of Aes Encryption With Ecc”, Proceedings of International Interdisciplinary Conference On Engineering Science & Management Held on 17th - 18th December 2016, in Goa, India.ISBN: 9788193137383.

[9] A. K. Mandal, C. Parakash, and A. Tiwari. "Performance evaluation of cryptographic algorithms: DES and AES."Electrical, Electronics and Computer Science (SCEECS), 2012 IEEE Students' Conference on. IEEE, pp 1-5.

[10] S. Sowmiya, I. Monica Tresa, A. Prabhu Chakkaravarthy, “Pixel based image encryption using magic square”, 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), 10.1109/ICAMMAET.2017.8186634.

[11] D.I. George Amalarethinam, J. Sai geetha, "Enhancing Security Level for Public Key Cryptosystem Using MRGA", World Congress on Computing and Communication Technologies (WCCCT), pp. 98-102, 2014, ISBN 978-1-4799-2876-7.

[12] Congxu Zhu ,2 and Zhijian Wang, “An Improved Piecewise Linear Chaotic Map Based Image Encryption Algorithm”, Volume 2014 |Article ID 275818

[13] Kaixin Jiao, Chen Pan, and Xiaoling Huang, “An Effective Framework for Chaotic Image Encryption Based on 3D Logistic Map”, Volume 2018 |Article ID 8402578

[14] Ye Yang, Shijie Xu, Chunqiang Yu, and Xianquan Zhang, “Image Encryption with Double Spiral Scans and Chaotic Maps”, Volume 2019 |Article ID 8694678

[15] MATLAB 2015a and 2017a Summary and documentation.