ISSN: 2278-4632
Vol-10 Issue-09 No.03 September 2020

Method of generating public key matrix and using it for image encryption

[1]SAMARENDRA SAMAL, *Gandhi Institute of Excellent Technocrats, Bhubaneswa, India*
[2]SUJATA MALLIK, *Padmashree Krutartha Acharya College of Engineering, Bargarh, Odisha, India*

***Abstract*—It is very difficult to find the inverse of a matrix in Galois field using standard matrix inversion algorithms. Hence, any block-based encryption process involving matrix as a key will take considerable amount of time for decryption. The inverse of a self-invertible matrix is the matrix itself. So, if these matrices are used for encryption, the computational time of the decryption algorithm reduces significantly. In this paper, a new method of generating self-invertible matrix is presented. In addition to this, a new method of generating sparse matrices based on a polynomial function and the process of inversion of this matrix without using standard matrix inversion algorithms is also presented. The product of these two types of matrices constitutes the public key matrix whereas the matrices individually act as the private keys. This matrix will have a large domain and can also be used to design an asymmetric encryption technique. The inverse of the key matrix can be calculated easily by the receiver provided the components of the key i.e. the self-invertible and the sparse matrices are known. This public key is used to encrypt images using standard image encryption algorithm and it is tested with various gray-scale images. After encryption, the images are found to be completely scrambled. The image encryption process has very low computational complexity which is evident from comparison with AES(128). Moreover, since the number of key matrices are huge, brute force attack becomes very difficult.**

I. INTRODUCTION

In the present era, the communication and storage of images has to be done in a secured manner due to the high confidentiality and importance of multimedia data. These data find widespread applications in cable-TV, online personal photograph album, medical imaging systems, military image communications, confidential video conferences, etc. Image encryption is one of the methods to ensure security of images[2].

The correlation among neighboring pixels of any natural image is very high. Thus, the value of any given pixel of an image can be predicted with reasonable precision from the values of its neighboring pixels. In order to reduce the correlation among pixels and increase the entropy, block based transformation algorithms are useful[4].

An image can be represented as a matrix of integers. In a block-based transformation algorithm, the image is divided into a number of blocks of equal sizes. Transform matrices are used to transform the image blocks[1], [3], [5], [6], [7]. For example, if an image E is represented as a $M \times N$ matrix of integers and P and Q are the transform matrices of dimension $M \times M$ and $N \times N$ respectively, E can be transformed into a matrix T of dimension $M \times N$ by the following relation.

$$T = PEQ \quad (1)$$

If P and Q are self-invertible matrices, then the original image can be recovered very easily. But, if the transform matrices are non-singular but not self-invertible, then E can be recovered by the following relation.

$$E = P^{-1}TQ^{-1} \quad (2)$$

The analyses presented here for generation of selfinvertible matrix are valid for matrix of positive integers, that are the residues of modulo arithmetic on a prime number i.e. all operations are carried out in Galois Field GF(p)[1], [8]. In GF(p), inverse of a number has to be found out by algorithms such as Extended Euclidean Algorithm. Any standard matrix inversion algorithm requires calculation of lot of inverses of numbers. Hence, inversion of high order matrices in Galois field involves a lot of computations and will take considerable amount of time. So, the motivation for this paper is to design matrices whose inverses can be calculated easily with knowledge of some secret parameters. Without knowledge of the secret parameters, the inverse of the matrices has to be calculated using standard algorithms. Self-invertible matrices(SIM) is one of these matrices. A matrix M is selfinvertible if $M.M = I$ i.e$M = M^{-1}$, where I is the identity matrix.

In Section II, a new method of generating SIM is presented. The rest of the paper is organized as the following. Method of generating sparse matrices based on a polynomial function and algorithm for finding its inverse is explained in Section III. In Section IV, the various key matrices that are used are shown and the algorithm of encryption and decryption is presented. Section V and VI provides the simulation result and cryptanalysis respectively. Section VII is the comparison of the proposed method with AES(128). Finally, the conclusion is presented in Section VIII.

II. PROPOSED METHOD OF GENERATING

ISSN: 2278-4632
Vol-10 Issue-09 No.03 September 2020

SELF-INVERTIBLE MATRICES

In this section, a novel method of generating self-invertible matrices is presented.

Let A be a matrix of size $(m \times m)$, where 'm' is any integer. If a matrix 'B' is represented in the following way

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

where B_{11}, B_{12}, B_{21} and B_{22} are the partition matrices generated using 'A' by the following relations

$$B_{11} = A^3$$
$$B_{22} = -A^3$$
$$B_{12} \times B_{21} = I - A^6$$

then, B will be self-invertible. Thus, B_{12} or B_{21} can be any factor of the expression $I - A^6$. By using this method, several self-invertible matrices can be generated. The proof is self evident.

Example:

Let 'A' be a 3 × 3 matrix generated randomly in GF(13). In this example all operations are carried out in GF(13).

Let

$$A = \begin{bmatrix} 3 & 6 & 5 \\ 2 & 9 & 10 \\ 4 & 7 & 11 \end{bmatrix}, \text{ then } A^3 = \begin{bmatrix} 12 & 0 & 1 \\ 6 & 6 & 9 \\ 4 & 6 & 8 \end{bmatrix}$$

In the following cases, different factors of $I - A^6$ are used to construct B_{12} and B_{21} such that (6) is satisfied.

Case I -

Let

$$B_{12} = I, \text{ then } B_{21} = I - A^6 = \begin{bmatrix} 9 & 7 & 6 \\ 12 & 2 & 11 \\ 1 & 7 & 9 \end{bmatrix}$$

$$\begin{bmatrix} 6 & 6 & 9 \\ 0\,1 & 0 & 12\,0\;1 \\ 1\,0 & 0 & \text{and } B = \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 4 & 6 & 8 \\ & 0\,0 & 1\,9 \\ & 7 & 6 \\ & 1\,0\,12 & \\ & 12\;2\;11\;7\;7 & 4 \\ 1 & 7 \quad 9 & 9\;7 \quad 5 \end{bmatrix}$$

Case II -

$$\text{and } B = \begin{bmatrix} 6 & 6\;9\;11\;5 & 6\;8\;9 & 6 \\ & = & & \\ & 4\;9 & & 2\;2 \\ 7 & 9\;2 & 7 \quad 7 & 4 \\ 7 & 0\;6 & 9 \quad 7 & 5 \end{bmatrix}$$

Case III -

Let $B_{12} = I - A$, then $B_{21} = (I + A)(I + A^2 + A^4)$,

$$12\;0\;1\;11\;7 \qquad 8$$

Let $B_{12} = I + A$, then $B_{21} = (I - A)(I + A^2 + A^4)$,

and

$$B = \begin{bmatrix} 12 & 0 & 1 & 4 & 6 & 5 \\ 6 & 6 & 9 & 2 & 10 & 10 \\ 4 & 6 & 8 & 4 & 7 & 12 \\ 12 & 6 & 10 & 1 & 0 & 12 \\ 6 & 1 & 3 & 7 & 7 & 4 \\ 11 & 11 & 0 & 9 & 7 & 5 \end{bmatrix} \quad (3)$$

Case IV -

Let $B_{12} = I - A^2$, then $B_{21} = (I + A^2 + A^4)$, (4) (5) (6)

$$\begin{bmatrix} 6 & 6 & 9 \\ 1 & 7 & 11\,12\;0 \\ 1\,12\,10\;0 & \text{and} & \end{bmatrix}$$

$$B = \begin{bmatrix} 4 & 6 & 8 & 8 & 5 & 11 \\ 4 & 4 & 6 & 1 & 0 & 12 \\ 0 & 5 & 9 & 7 & 7 & 4 \\ 9 & 12 & 3 & 9 & 7 & 5 \end{bmatrix}$$

Case V -

$$\begin{bmatrix} 64 & 6\,9\,7\,86\,8\,9\,7 & 46 \\ B = & 064 \quad 0\,1\,1\,0\,12\,7\,9\,7 & \text{and} \\ & 76\,9\,9\,7 \quad 45 & \end{bmatrix}$$

Case VI -

Let $B_{12} = I - A^3$, then $B_{21} = (I + A^3)$,

$$12\;0\;1\;2\;0\;12$$

Let $B_{12} = (I - A)(I - A + A^2)$, then $B_{21} = (I + A + A^2)$,

$$\begin{bmatrix} 6 & 6 & 9 \\ 1 & 4 & \\ 1\,12 & 0 & \\ 1 & 0 & \\ 7 & 2 & \text{and } B = \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

$$\begin{bmatrix} 4 & 6 & 8 & 11\,9 \\ 3 & & & \\ 10 & 5 & 11 & 1 \quad 0\,12 \\ 8 & 0 & 7 & 7 \quad 7 \quad 4 \\ 9 & 10\,11 & 9 & 7 \quad 5 \end{bmatrix}$$

All other cases are transpose of all the matrices cited above. This method can also be extended by taking any value of n, i.e. by assuming $B_{11} = A^n$, for $n > 3$.

III. GENERATION OF SPARSE MATRIX

An eigen function can be used to generate a sparse matrix.

If $f(\lambda) = \lambda^n + s_{n-1}\lambda^{(n-1)} + \cdots + s_0 = 0$ is an eigen function, its corresponding sparse matrix 'S' can be presented as below.

ISSN: 2278-4632
Vol-10 Issue-09 No.03 September 2020

$$S = \begin{bmatrix} -s_{n-1} & 1 & 0 & 0 & 0 & \cdots & 0 \\ -s_{n-2} & 0 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ -s_1 & 0 & 0 & 0 & 0 & \cdots & 1 \\ -s_0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

Different eigen functions can be obtained by changing the coefficients of the function thereby resulting in different sparse matrices.

A. Method to find the inverse of sparse matrix using eigen function

The inverse of the matrix of above form can be found out by the following method without using standard matrix inversion algorithms provided the eigen function is known. Let 'T' is the inverse of the above sparse matrix 'S' and is represented as below..

$$T = \begin{bmatrix} 0 & 0 & \cdots & 0 & t_{n-1} \\ 1 & 0 & \cdots & 0 & t_{n-2} \\ 0 & 1 & \cdots & 0 & t_{n-3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & t_0 \end{bmatrix}$$

Since, $T = S^{-1} \Longrightarrow S.T = I$ and using this relation, the elements of 'T' can be found out in the following way.

$$\begin{aligned} -s_{n-1}t_{n-1} + t_{n-2} &= 0 \\ -s_{n-2}t_{n-1} + t_{n-3} &= 0 \\ -s_{n-3}t_{n-1} + t_{n-4} &= 0 \\ \cdots\cdots\cdots\cdots\cdots & \\ -s_1 t_{n-1} + t_0 &= 0 \end{aligned}$$

(9)
(10)
(11)
(12) and $-s_0 t_{n-1} = 1$ (13) From equation (13)

$$t_{n-1} = \left(\frac{-1}{s_{n-1}}\right)$$

Substituting $t_{n-1} = \left(\frac{-1}{s_{n-1}}\right)$ in rest of the equations $t_0 = \left(\frac{-s_1}{s_{n-1}}\right)$, $t_1 = \left(\frac{-s_2}{s_{n-1}}\right) \cdots t_{n-3} = \left(\frac{-s_{n-2}}{s_{n-1}}\right)$ and $t_{n-} \left(\frac{-s_{n-1}}{s_{n-1}}\right)$

IV. Generation of Key Matrices and Algorithm of Encryption and Decryption

Let 'A' be a 8 × 8 self-invertible matrix generated by the method discussed in section II. The eigen functions $\lambda - 31, \lambda - 7, \lambda - 191, \lambda^3 + \lambda^2 + 1, \lambda - 47$ and $\lambda - 127$ are used to generate a 8 × 8 sparse matrix 'B' by the method discussed in section III(Case - V). These eigen functions are used to populate the diagonal elements of the matrix. The rest of the elements are zero.

A. Key matrices

The key matrices that are used in the simulation are presented below.

226 218 151 15926 33 100
92 159 181 91 107 92
71 160 144
243 96 199 1098 155
53 142 12282 209 44
129 16942 208

(7)

$A =$

227 218 151 159 25 33 100 92
159 182 91 107 92 70
160 144 243 96 200 1098
155 52 142
122 82 209 45 129 169 42 207
31 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0

$B =$

0 0 191 0 0 0 0 0 0 0 0 1 1 0
0 0 0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 47
0
0 0 0 0 0 0 0 127

(8)

The public key matrix 'C' is generated by the relation C = A.B.A

45 95 198 73 140 150 16 25 147 22
181 104 147 57 165 240

$C =$

29 202 104 205 70 228 153 74
98 196 173 107 23 97 9 188
68 150 78 25 213 4 88 12 201 214 113
70 210 173 81 27
129 44 100 120 129 189 161 156 8 63
52 165 12 56 77 79

The inverse of the public key matrix i.e. 'D' is generated by the relation $D = A.B^{-1}.A$

131 224 48 1 225 161 127 133 106
184 221 181 25 41 145 248

$D =$

158 189 81 212 65 197 6 44
88 195 40 103 146 52 38 234 185 58 41
174 104 131 187 157
4 132 177 54 138 51 136 10
213 245 225 122 180 149 43 173
221 146 173 191 226 131 133 120

B. Algorithm of encryption

Step 1. Divide a gray-scale image into 8 × 8 blocks.
Step 2. The public key matrix 'C' is generated as discussed in subsection IV-A.
Step 3. Encrypt each block of the image using the public key.
Step 4. Combine all the blocks to form the encrypted image.

C. Algorithm of decryption

Step 1. Divide the encrypted image into 8 × 8 blocks.
Step 2. Find the inverse of 'B' by the method discussed in subsection III-A.
Step 3. Generate the decryption key matrix 'D' as discussed in subsection IV-A.

ISSN: 2278-4632
Vol-10 Issue-09 No.03 September 2020

Step 4. Decrypt each block of the encrypted image using 'D'.

Step 5. Combine the decrypted blocks to get back the original image.

V. RESULTS

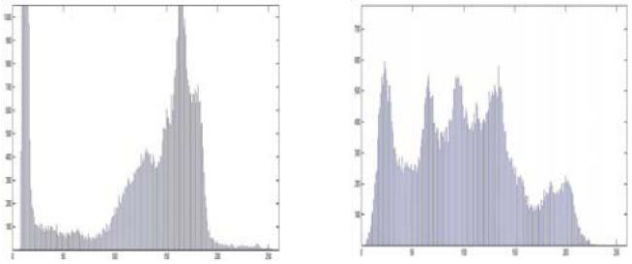
Cameraman and Lena images of 256 × 256 resolution were used for simulation of the algorithm. The original images and their histograms are shown in Fig. 1 and 2 respectively.

(a) Cameraman (b) Lena

Fig. 1: Original image

The original images were rounded to 251 pixels as encryption is done in GF(251) and are shown in Fig. 3. The corresponding histograms are shown in Fig. 4.

The public key 'C' is used to encrypt the original images (rounded to 251 pixels) to get the encrypted images which are shown in Fig. 5. The corresponding histograms are shown in Fig. 6.

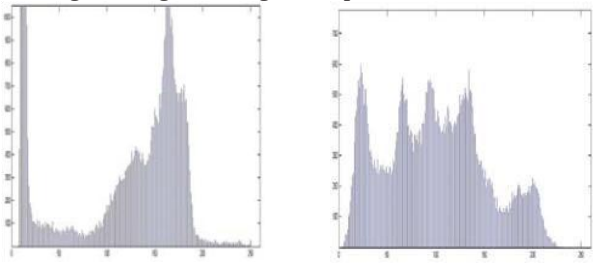


(a) Cameraman (b) Lena

Fig. 2: Histograms of original image

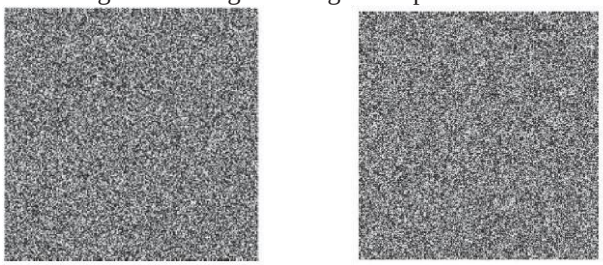
(a) Cameraman (b) Lena

Fig. 3: Original image with pixels ≤ 251



(a) Cameraman (b) Lena

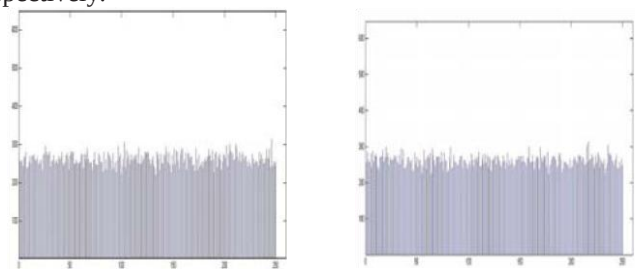
Fig. 4: Histograms of original image with pixels ≤ 251



(a) Cameraman (b) Lena

Fig. 5: Encrypted image

The decryption matrix 'D' is used to get back the original images (rounded to 251 pixels). The decrypted images and their histograms are shown in Fig. 7 and 8 respectively.

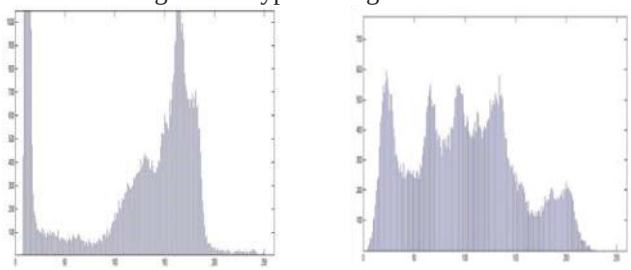


(a) Cameraman (b) Lena

Fig. 6: Histograms of encrypted image

(a) Cameraman (b) Lena

Fig. 7: Decrypted image



(a) Cameraman (b) Lena

Fig. 8: Histograms of decrypted image

VI. CRYPTANALYSIS

After encryption, the image gets completely scrambled and no trace of the original image remains in it. This can be seen in Fig. 5 of the simulation result. Statistical attack is very difficult as all the pixels have nearly identical frequencies in the encrypted image. This is evident from the histograms of the encrypted image shown in Fig. 6.

ISSN: 2278-4632
Vol-10 Issue-09 No.03 September 2020

The number of n × n sparse matrices in GF(p) that can be generated from a single eigen function $f(\lambda) = \lambda^n + s_{n-1}\lambda^{(n-1)} + \cdots + s_0 = 0$ is equal to p^n. For $p = 251$ and $n = 8$, this is nearly equal to 2^{64}, whereas for $p = 251$ and $n = 16$, the number of sparse matrices is nearly equal to 2^{128}. Thus, the intruder has to test a huge number of key matrices to get the private key 'B'.

Number of n × n self-invertible matrices that exist in GF(p) can be analytically found out to be $\frac{(p-2)^n}{2}$. For $p = 251$ and $n = 8$, the number of self-invertible matrices is equal to $\frac{249}{2^8}$ whereas for $\underset{16}{p} = 251$ and $n = 16$, the number of such matrices is equal to $\frac{249}{2}$ which is quite large.

From the above analysis, it can be observed that if the image is divided into 8 × 8 blocks and encryption is carried out in GF(251), the intruder has to test $251^8 \times \frac{249^8}{2} \approx 2^{128} \approx$the number of possible keys in AES(128). Thus, the attacker has to make a large number of trials in order to find the private key matrices by brute force attack. Moreover, when the block size increases, the number of keys increases exponentially.

A. Avalanche Effect

Analysis of avalanche effect(AE) over an algorithm gives an idea about the diffusion property of the technique. Avalanche effect can be calculated as the ratio of number of bits flipped in the ciphered image with respect to the total number of pixels when a single bit of any random pixel of the original image is complemented. The desired value of avalanche effect is 50. The proposed algorithm is tested with irisflower, barbara and lena images of resolution 256 × 256, 512 × 512 and 1024 × 1024 respectively and different values of AE are shown in table I.

TABLE I: Avalanche effect(AE)

Image	Resolution	AE
Iris flower	256 × 256	49.85
Barbara	512 × 512	50.04
Lena	1024 × 1024	50.02

VII. COMPARISON OF PROPOSED ALGORITHM WITH AES(128)

Irisflower, barbara and lena images of resolution 256 × 256, 512 × 512 and 1024 × 1024 respectively were tested with AES(128bit key) and the proposed algorithm. These standard images are shown in Fig. 9.

In table II, the entropy of the original images are shown. Two properties of an image i.e entropy(E) and 2D correlation coefficient(C) can give an idea of the degree of interdependence among neighboring pixels. A higher entropy signifies a greater degree of randomness among the pixels whereas a low 2D correlation coefficient's value is an indicator of variance of the pixels of the encrypted image with respect to the original image. The different images are tested with the two algorithms under identical conditions. The approximate execution time(T),

(a) Iris flower (b) Barbara (c) Lena

Fig. 9: Standard images used for comparison of proposed algorithm with AES(128)

the entropy(E) of the encrypted image and the 2D correlation coefficient(C) of the encrypted images are shown in table III and IV for the proposed algorithm and AES(128) algorithm respectively.

Number of pixels change rate (NPCR) is defined as the rate of change of the number of pixels of the cipher image when only one pixel of the plain image is modified. The unified average changing intensity (UACI)measures the average intensity of differences between the plain image and ciphered image [9]. These 2 parameters are measured in terms of C_1 and C_2 where C_1 is obtained by encryption of original image and C_2 is obtained by encryption of modified original image (change of any 1 random pixel of original image). A high NPCR value and a low UACI value is an indicator of robustness against differential attacks. Table V and VI shows the simulation result of NPCR and UACI analysis respectively.

TABLE II: Entropy of Original Images

Image	Resolution	Entropy
Iris flower	256 × 256	7.6958
Barbara	512 × 512	7.4664
Lena	1024 × 1024	7.5992

TABLE III: Results of encryption by proposed algorithm

Image	Proposed algorithm		
	E	C	T(sec)
Iris flower	7.9684	0.0084	0.209668
Barbara	7.9707	0.002297	0.400325
Lena	7.9714	0.000864	0.960873

TABLE IV: Results of encryption by AES(128) algorithm

Image	AES(128)		
	E	C	T(sec)
Iris flower	7.9972	0.004737	69.8727
Barbara	7.9992	0.001992	278.67642
Lena	7.9998	0.002453	1171.609359

TABLE V: NPCR analysis

Image	Proposed Algorithm	AES(128)
Iris flower	0.99638	0.99589
Barbara	0.99605	0.99591
Lena	0.99604	0.99613

ISSN: 2278-4632
Vol-10 Issue-09 No.03 September 2020

TABLE VI: UACI analysis

Image	Proposed Algorithm	AES(128)
Iris flower	0.32724	0.33458
Barbara	0.32868	0.33477
Lena	0.328	0.33415

VIII. CONCLUSION

In this paper, a new method of generating self-invertible matrices is presented and an asymmetric image encryption technique is designed using self-invertible matrix as part of the public key. The computational and time complexity of the decryption algorithm decreases significantly as there is no need to invert the public key matrix using standard matrix inversion algorithms. But the intruder has no such advantage as he/she has no knowledge of the private keys ('A' and 'B') which is necessary to invert the public key without using matrix inversion algorithms. Moreover, if the modulo value is taken as a large number, the number of self-invertible matrices becomes very large thereby thwarting brute-force attack.

The robustness of the algorithm is evident from the cryptanalysis as well as the NPCR and UACI analyses. The NPCR values for almost all the figures considered are higher for the proposed algorithm in comparison to AES(128). Also, UACI values are lower in comparison to AES(128). The entropy and 2D correlation coefficient's value of the encrypted images vary marginally in comparison to AES(128) whereas the approximate execution time of the proposed method is significantly lower, thereby justifying the claim of low computational and time complexity.

REFERENCES

[1] S.K. Chhotaray, AnimeshChhotaray, G.S. Rath, "Orthonormal matrices and image encryption", *International Conference on Circuits, Devices and Communication*, BIT Mesra, 12-13 September 2014.

[2] Komal D Patel, SonalBelani, "Image Encryption Using Different Techniques:A Review", *International Journal of Emerging Technology and Advanced Engineering*, Vol. 1, Issue 1, November 2011.

[3] Delong Cui, Lei Shu, Yuanfang Chen, Xiaoling Wu, "Image encryption using block-based transformation with fractional Fourier transform", *8th International ICST Conference on Communications and Networking*, pp. 552 - 556, 14-16 August 2013.

[4] Mohammad Ali Bani Younes , Aman Jantan, "Image encryption using block-based transformation algorithm", *IAENG International Journal of Computer Science*, 35:1, *IJCS*35_1_03, Advance online publication: 19 February 2008.

[5] Karagodin M.A., Osokin A.N., "Image compression by means of walsh transforms", *IEEE Trans. on Modern Techniques and Technologies*, pp. 173 - 175, 2002.

[6] K.R. Rao, N. Ahmed, "Orthogonal transforms for digital signal processing", *IEEE International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, USA, Vol. 1, pp. 136 - 140, 12-14 April 1976.

[7] Williams K. Pratt, Julius Kane, Harry C. Andrews, "Hadamard transform Image Coding", *Proceedings of the IEEE*, Vol. 57, pp. 58 - 68, 1969.

[8] B. Acharya, G.S. Rath, S.K. Patra, S.K. Panigrahy, "Novel methods of generating self-invertible matrix for hill cipher algorithm", *International Journal of Security*, Vol. 1, Issue 1, pp. 14 - 21.

[9] H. Khanzadi, M. Esghi, S.E. Borujeni, "Image Encryption Using Random Bit Sequence Based on Chaotic Maps", *Arabian Journal for Science and Engineering*, Vol. 39, Issue 2, pp. 1039 - 1047,2014.