Image Encryption Using Pseudorandom Permutation

Tapan Kumar Hazra, Kishlay Raj, M. Sumanth Kumar, Soummyo Priyo Chattopadhyay and Ajoy Kumar Chakraborty

Abstract A simple, fast, and dynamic image encryption scheme is proposed in this paper based on pixel shuffling. The primary idea applied for the purpose lies in designing of dynamic pseudorandom permutation map using simple divide and conquer method that will introduce diffusion among strongly correlated image pixels. As a result, the visual information content is completely lost. Data encryption is to hide and conceal the information content present in the secret data. Data encryption differs from data hiding, where the prime objective is to conceal the secret data. Using the technique of reversible data encryption, we can get back the original data content out of the encrypted message, without loss of any original information. The decryption process can be applied only at the authentic receiver end. Thus, we can transfer the encrypted data through any communication channel because it has completely lost its resemblance to the original data. Though theoretically it is possible to break such security, it does not appear feasible by any known practical means as it appears as random noise to any unintended recipient. To incorporate second fold of security, some proper data hiding methods can be used to embed the encrypted data. In this paper, we have presented a new visual data encryption technique by randomly shuffling the pixels within the image. The newly formed encrypted image using this technique completely loses the original characteristics. Our experimental results show that the proposed method can achieve the claim.

Keywords Circular left shift · Circular right shift · Cryptography
Divide and conquer · Horizontal encryption · Image encryption
Pseudorandom permutation · Vertical encryption

T. K. Hazra (✉) · K. Raj · M. Sumanth Kumar · S. P. Chattopadhyay · A. K. Chakraborty
Department of Information Technology, Institute of Engineering & Management, Y-12, Salt Lake Electronics Complex, Sector-V, Kolkata, India
e-mail: tapankumar.hazra@iemcal.com; tapankumarh@yahoo.com

M. Chakraborty et al. (eds.), *Proceedings of International Ethical Hacking Conference 2018*, Advances in Intelligent Systems and Computing 811,
https://doi.org/10.1007/978-981-13-1544-2_11

1 Introduction

It is Internet and dawn of the digital natives that have made this world a small village with advanced information and communication system for the society. The whole of the global system is intensifying with the quantity of data that is stored and transmitted. Unfortunately, this ease has also raised new challenges concerning the security and protection of important information and data against unauthorized access. Increase in interconnectivity, growth of networks, and number of users and decentralization has increased system vulnerability. So the much-needed security of information and communication system involves the protection and confidentiality of the system and the data to be transmitted and stored.

Steganography and cryptography are two different information and data hiding techniques. Steganography hides messages inside some other digital media while cryptography protects the content of messages for secure communication in the presence of third parties (called adversaries) [1–4]. Sometimes both reapplied in succession [5, 6]. The word steganography is derived from Greek, and it means "covered writing", the art of hiding information in ways that prevent detection. There can be many ways to hide information. To embed secret data, either every byte of cover media are selected sequentially, or bytes are selectively chosen from insignificant regions that draw less attention of viewer [5]. After location selection in cover media, a straight message insertion may encode every byte. Messages may also be scattered following some SCAN pattern throughout the cover data [6]. Least significant bit (LSB) insertion is one of the common methods of data hiding.

Image encryption techniques have been increasingly studied to satisfy the demands for secure image transmission over the network. Traditional image encryption techniques which are nonrandom and static in nature are more vulnerable to be hacked with possibility of finding similarity in the encoded image. In most cases, the encoded message that comes out is same every time we apply it on the same image. In this paper, we present an image encryption technique which is both dynamic and robust. It generates a new random image with set of keys, every time we apply it on an image, making it pseudorandom and at the same time difficult to analyze or to decode. We have found that the encryption technique discussed here creates new random images, out of which none of them carry any resemblance with the original image and seem to be pure noise.

In the present paper, we have considered the visual information in a grayscale image and propose a new method by shuffling pixel information randomly within the image. The technique breaks the array of pixels of the image into different subsets using divide and conquer technique, applied in a way similar to merge sort. Then a circular rotation is performed on every subset after dividing the array with the help of a random key generated. This way of shuffling pixel information within the image dimension gives some sort of pseudorandom permutation and also we can get back the original image exactly using generated key.

The organization of the paper is as follows: Sect. 2 focusses on existing works on image encryption, Sect. 3 describes the proposed encryption and decryption algorithm with illustrations, Sect. 4 focusses on result and discussion, and Sect. 5 concludes the paper.

2 Existing Works on Image Encryption

Digital image is a two-dimensional array of pixel intensities, and those intensity values are highly correlated for plain image. Image encryption techniques primarily disrupt the normal interpretation of visual information represented by those intensity values. There are various methods to achieve this. Pseudorandom permutation is applied to shuffle spatial locations of pixels within image boundaries. Using pseudorandom permutation (PRP), we can generate permutation cipher that is successfully applied on text or images to perform encryption [1–3]. The main limitation is this cipher is vulnerable to statistical attack, and security increases with the length of the key. In case of text file encryption, slightly different techniques are applied [7, 8].

Image encryption based on chaos theory is very popular and efficiently encrypt images. Limitations of PRP-based image encryption are mostly overcame [9–17].

Image encryption using principle of optics is also applied [18]. There is also some research work that focusses on security enhancement in addition to standard RSA algorithm [19, 20].

2.1 Pseudorandom Permutation

In the present work, we have rearranged the information using divide and conquer and circular right shift technique which creates a new randomly rearranged image of the original image, every time we apply it. The proposed technique is not truly random and generates a set of keys, or else it would become impossible to get back the original data.

We applied the proposed technique on image data which are intensity level of pixels. We treat the image as a simple 2-D array with pixels intensity values. This technique works on each line of pixels treating them as simple 1-D array.

At each step in the process, we break the array in two equal halves (ignore the middle element in case of odd number of elements). We swap the positions of each of the elements to the other side. Now we do the right shift of the each of these two groups for which we generate two random numbers and perform the right shift with these values. The randomly generated numbers are stored in the key set where it will be used in the future to retrieve the information back. The steps are repeated recursively until we break them down to the level of individual pixels.

3 Proposed Algorithm

A digital image is a 2-D array of pixel intensity values. The proposed algorithm consists of two parts: encryption and decryption. Encryption process is done by shuffling the elements horizontally row-wise as well as vertically column-wise. We call these as horizontal encryption and vertical encryption, respectively. Exactly inverse operations are done in decryption.

3.1 Encryption Process

This technique works on each row of pixels, treating them as simple 1-D array. We applied the discussed technique on an image data to the level of pixels. We have rearranged the information using divide and conquer and circular right shift technique which creates a different randomly rearranged image of the original image, every time we apply it.

At each step in the process, we break the array in two equal halves (ignore the middle element in case of odd number of elements). We swap the positions of each of the elements to the other side. Now we do the right shift of the each of these two groups for which we generate two random numbers and perform the right shift with these values. The randomly generated numbers are stored in the key set where it will be used in the future to retrieve the information back. The steps are repeated recursively until we break them down to the level of individual pixels.

For example, let us take an array of 10 elements consisting of numbers from 0 to 9, and each encryption algorithm steps are specified and illustrated with the help of these dummy elements.

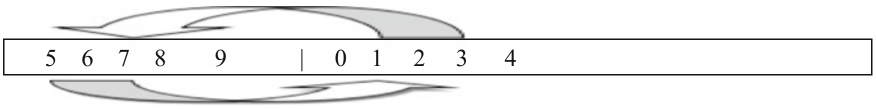
0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9

Step 1: Break into two equal parts (divide).

0 1 2 3 4	5 6 7 8 9

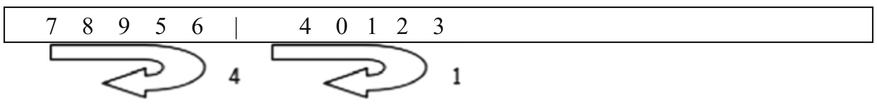
Step 2: Swap two parts



5 6 7 8 9	0 1 2 3 4

Step 3: (a) Generate two random numbers
Let random number 1 is 4 and random number 2 is 1

(b) Perform Circular Right Shift on each half by respective random number.



Step 4: Break each part into two sub parts and apply steps 1 through 3 recursively until number of elements in each part is two.

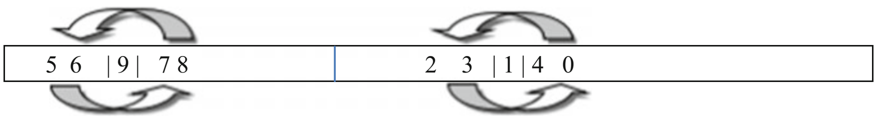
Further illustrations for successive recursive calls are shown in sub-steps and associated diagrams as follows:

Step 1:

78|9|56 | 40|1|23

Step 2:

Swap each sub-part



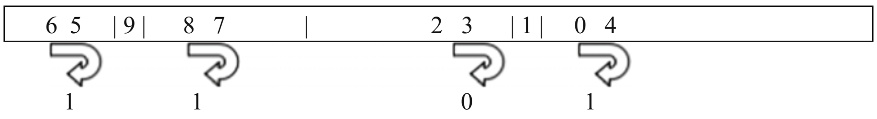
Step 3:

Generate random numbers for each part:

For first part, random number 1 is 1 and random number 2 is 1.

For second part, random number 1 is 0 and random number 2 is 1.

Perform circular right shift



Now stop the recursive calls as the number of elements in no exceeding 2. So the final encrypted array obtained is

6 5 9 8 7 2 3 1 0 4

And the randomly generated key is generated as **4 1 1 1 0 1**.

This encryption technique is applied on each row of the 2-D array of pixels treating them as separate 1-D array.

Now, the same technique can also be applied to the vertical line, i.e., column-wise, treating them as another form of 1-D array.

We can apply the technique both horizontally and vertically (every time getting a more complex shuffled image), and therefore, we need to remember the sequence and do the exact reverse to get back the original image.

Though the technique seems to be relatively simple but gives great results with generation of a new random image, every time we apply it. Each encryption will produce completely different key, and size of the key depends on input matrix.

This technique is also very flexible as well because any user who wants to encrypt can create his/her own sequence of horizontal and vertical encryption which will have a relatively different technique to decode (the exact reverse sequence which only the user knows). So it can be easily customized and still remains utterly difficult to decode.

To understand its functioning more deeply, let us take an example of a simple 10 × 10 matrix

$$\begin{array}{c}
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\
0\,1\,2\,3\,4\,5\,6\,7\,8\,9
\end{array}$$

Now we apply to above-discussed method on the first row of matrix and that becomes

7 6 5 8 9 4 3 2 0 1 <--- encryption applied

$$\left.\begin{array}{l}
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\\
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\\
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\\
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\\
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\\
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\\
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\\
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\\
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9
\end{array}\right\} \text{others rows are not disturbed}$$

With the key 2 0 0 1 0 1 for row 1, which were again generated randomly so the permutation came out to be different from that of the previous case. We now apply the same technique to all the rows of the given matrix.

	Row
7 6 5 8 9 4 3 2 0 1	1
9 8 7 5 6 0 4 3 1 2	2
6 5 9 7 8 0 4 3 1 2	3
6 5 9 7 8 0 4 3 2 1	4
8 7 6 5 9 1 0 4 3 2	5
6 5 9 7 8 1 0 4 2 3	6
8 9 7 5 6 3 4 2 0 1	7
9 5 8 7 6 0 1 4 3 2	8
6 7 5 8 9 0 1 4 2 3	9
8 9 7 6 5 2 3 1 0 4	10

We call this part of encryption as horizontal encryption. Horizontal encryption jumbles up all the rows among themselves. We also keep the keys generated for encryption in 2-D array form so that we can distinguish keys for different rows easily. So, the list of keys in 2-D array form is given below.

Keys	
2 0 0 1 0 1	for Row1
0 4 0 1 0 1	for Row2
3 4 0 1 0 1	for Row3
3 4 0 1 0 1	for Row4
1 3 0 0 0 0	for Row5
3 3 0 1 0 1	for Row6
0 0 1 1 1 1	for Row7
4 3 1 0 1 0	for Row8
2 3 1 1 1 1	for Row9
0 1 1 0 1 0	for Row10

To make our encryption more secure, we apply the same method on the vertical lines, i.e., on the columns of the encrypted array and treating each column as single 2-D array. We call the method of applying encryption on each vertical lines, i.e., columns as vertical encryption. Separate key is produced dynamically for the process.

So, we observe that just after two series of encryption one after other the encrypted matrix has lost its resemblance totally with the original matrix. Unlike other transposition ciphering processes, the proposed process is dynamic and random, and localized transposition effect is absolutely removed for large 2-D array.

To get back the original matrix, we have to perform the decryption technique in the exact reverse manner that we choose to encrypt it. In this situation, we first performed horizontal encryption and then the vertical encryption on it. So we will have to first perform the vertical decryption on it and then the horizontal decryption.

When we perform the vertical decryption, we get back the matrix which was formed after the horizontal encryption was applied to the original matrix so as second step we perform horizontal decryption and get back the original matrix.

```
Input: Original Matrix
Step1: Horizontal encryption (on rows)
Step2: Vertical encryption (on columns)
Step3: Vertical decryption
Step4: Horizontal decryption
```

Any other sequence of decryption other than the exact reverse of the original process will not be able to recover the original matrix.

This also means the encryption technique can be easily customized by user making it more secure, because the sequence will be known only to the original key. For example, if the encryption was performed via three successive horizontal encryptions, only three horizontal decryptions will be able to decrypt it with the corresponding keys.

```
Input: Original Matrix
Step1: Horizontal encryption (1)
Step2: Horizontal encryption (2)
Step3: Horizontal encryption (3)
Step4: Horizontal decryption (key set 3)
Step5: Horizontal decryption (key set 2)
Step6: Horizontal decryption (key set 1)
```

3.2 Decryption Process

We take the encrypted array 6 5 9 8 7 2 3 1 0 4 and break it down to the lowest level.

```
Step 1: Break into two parts
```

```
  6 5 9 8 7        |       2 3 1 0 4
```

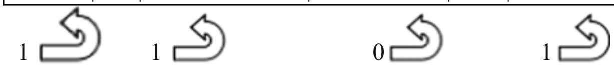
```
Again break it into sub parts
```

```
  6 5  | 9 | 8 7        |       2 3 | 1 | 0 4
```

Now, after reaching the lowest level, we will start using the key (randomly generated keys which were generated during the time of encryption). Key: 4 1 1 1 0 1.

We will start using the keys in exact reverse fashion and do circular left shift and then swap the elements.

6 5 | 9 | 8 7 | 2 3 | 1 | 0 4



After the circular left shift, the array becomes

5 6 | 9 | 7 8 | 2 3 | 1 | 4 0

Step 2: Swap each subpart

7 8 |9| 5 6 | 4 0 | 1 | 2 3

After swapping, the array becomes

7 8 9 5 6 | 4 0 1 2 3

Key: 4 1 1 1 0 1, and the part yet to used (using the reverse order) are 4 1

7 8 9 5 6 | 4 0 1 2 3

After the shift the array becomes

5 6 7 8 9 | 0 1 2 3 4

Performing the swap operation:

5 6 7 8 9 | 0 1 2 3 4

Decrypted array becomes

0 1 2 3 4 | 5 6 7 8 9

So, the shuffled elements are placed back to the original position correctly.

4 Results and Discussion

A digital image records pixels intensity values within a 2-D array of 8-bit unsigned integers. We applied the proposed encryption technique to the image and observed that the image has totally lost its resemblance to the original image. First encryption is applied on all the rows (horizontal lines). After encryption is completed on all the horizontal lines, we applied it to all the columns (vertical lines).

Even with the simplest permutation of horizontal and vertical encryption, the results were astounding. The technique is applied on Lena and Cameraman and results are shown in Figs. 1 and 2. Two different encryption results are shown, and various encryption quality parameters indicate that the proposed process is dynamic. We also found that the bigger the image (matrix) was, the better was the encryption because of the bigger right circular rotations possible during the encryption.

Necessary security analysis was carried out on plain image and encrypted images using the new algorithm, and simulation results show that encryption and decryption are good, and the algorithm demands good security and robustness. Table 1 represents entropy and PSNR values computed at various stages of encryption. Plain image Lena has entropy 5.332146228071736, and cameraman has entropy 5.037729765949852.

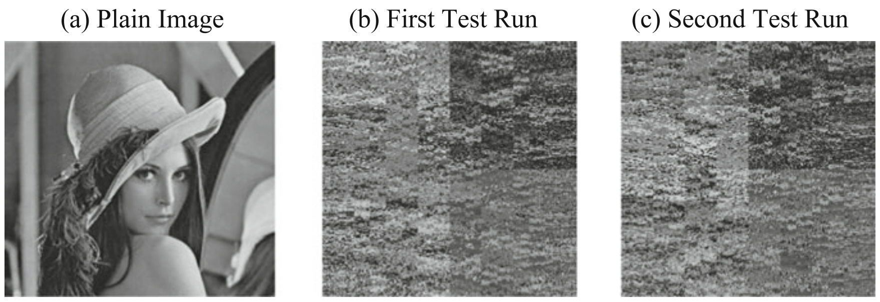


Fig. 1 Lena plain image and two encrypted image in two test runs

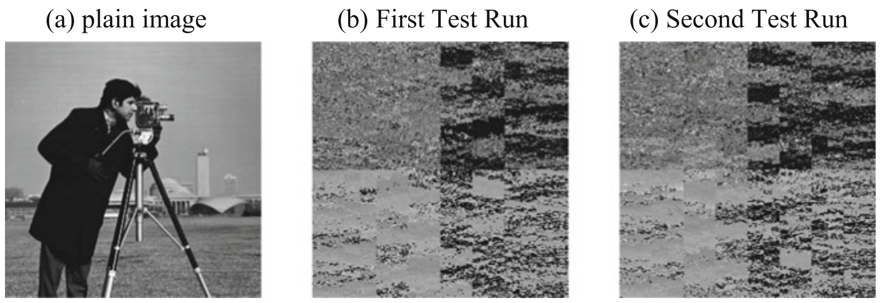


Fig. 2 Cameraman plain image and two encrypted image in two test runs

Table 1 Entropy and PSNR of encrypted images

Selected image	Reference image	PSNR of selected image	Entropy of selected image
Encrypted Lena, Fig. 1b	Lena plain image, Fig. 1a	−36.928817	5.332146228071736
Encrypted Lena, Fig. 1c	Lena plain image, Fig. 1a	−36.943271	5.332146228071736
Encrypted Cameraman, Fig. 2b	Cameraman plain image, Fig. 2a	−39.636006	5.037729765949852
Cameraman Second encrypt in Fig. 2c	Cameraman plain image, Fig. 2a	−39.673154	05.037729765949852

Table 2 Pearson's correlation coefficient

Selected image	Horizontal correlation coefficient	Vertical correlation coefficient
Plain image Lena, Fig. 1a	0.9071539397408981	0.9746624329061755
Encrypted Lena, Fig. 1b	0.3902039586852926	0.164632642344182
Encrypted Lena, Fig. 1c	0.4032467558705724	0.294682842482829
Plain image Cameraman, Fig. 2a	0.9433007313117971	0.8533010814343324
Encrypted Cameraman, Fig. 2b	0.49107879669398824	0.08674830224963347
Encrypted Cameraman, Fig. 2c	0.5031422316833177	0.32548714931357287

Pearson's correlation coefficient is computed using Eq. (1) and presented in Table 2 to determine the degree of correlation between adjacent pixels in both horizontal and vertical directions for all images. It is clearly observed that pixels of plain image are highly correlated, but the same is lost when encrypted by the proposed algorithm.

$$r = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right)\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2\right)}}, \quad \text{where} \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ and } \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{1}$$

Here, $(x_{i,}\ y_i)$ refers to the ith pair of vertically/horizontally adjacent pixels. Considering an image of dimensions $(a \times b)$, the value of n is set as, $n = (b - 1) * a$, in order to calculate the correlation coefficient for the horizontally adjacent pixels and $n = (a - 1) * b$.

Two of the most common techniques to evaluate the strength of image encryption algorithms, especially with respect to differential attacks, are NPCR (number of pixel changing rate) and UACI (Unified Average Changing Intensity). NPCR and

Table 3 Correlation coefficient

Referred images	Inter correlation coefficient	NPCR (%)	UACI (%)
Figure 1a, b	−0.07608793056431	99.5254516601563	22.5088560814951
Figure 1a, c	−0.079675319256034	99.5269775390625	22.5448907590380
Figure 1b, c	0.085660634687198	99.3209838867188	20.3426226447610
Figure 2a, b	−0.183095212288852	99.5651245117188	31.0831346698836
Figure 2a, c	−0.1933	99.6002197265625	31.3529818665748
Figure 2b, c	0.268571041672525	97.7081298828125	19.8884253408395

UACI are used to quantify the influence of change of pixel positions in the plain image. NPCR measures the number of pixel change rate, whereas UACI measures the average intensity of difference between the cipher image and the plain image.

The respective values of NPCR and UACI for all cases are computed using Eqs. (2) and (3) and are provided in Table 3. A high NPCR/UACI value indicates a high resistance to differential attacks.

$$NPCR : N\left(Cipher^{o}, Cipher^{c}\right) = \sum_{i,j} \frac{D(i,j)}{T} \times 100\,\% \tag{2}$$

$$UACI : u\left(Cipher^{o}, Cipher^{c}\right) = \sum_{i,j} \frac{[Cipher^{o}(i,j) - Cipher^{c}(i,j)\}}{F \times T} \times 100\,\% \tag{3}$$

where

$$D(i,j) = \begin{cases} 0, \ if\ Cipher^{o}(i,j) = Cipher^{c}(i,j) \\ 1, \ if\ Cipher^{o}(i,j) \neq Cipher^{c}(i,j) \end{cases}$$

The encryption algorithm proposed in the paper has the ability to resist brute-force attack. Encryption key is totally randomly generated.

Estimation of key size: Let the number of elements in one row/column is n. The size of the key involved in that row/column is decided by the number of random numbers involved to perform rotations. Every time the array is divided into two equal parts, skipping middle element in case of odd size, and random rotation is performed if length of subpart is more than one. So there are two random numbers at first stage, 2^2 in the 2nd stage, and so on until kth stage such that $\lfloor \frac{n}{2^k} \rfloor = 2$ i.e. $k = \lfloor \log_2 n \rfloor - 1$. So, the size of key = 2 + 2^2 + ⋯ + 2^{k} = 2^{k+1} – 2. For 2-D array, same key size is applied for each row/column. So exponential time is required for brute-force attacks.

With the key, the attacker also needs to know the proper sequence (horizontal or vertical) in which the encryption was made.

5 Conclusion

The proposed shuffle-based image encryption technique produces a dynamic result for each run of the application due to the inclusion of random number to achieve pseudorandom permutation of image pixels positions. Although the histogram remains same and computed entropy varies marginally, security can be breached for known images. But the proposed technique promises many advantages like large key space which is dynamic, and the process is fast. So the proposed technique may be a potential step in hybrid cryptosystem to overcome all existing limitations.

References

1. Hazra, T.K., Bhattacharyya, S.: Image encryption by blockwise pixel shuffling using modified Fisher-Yates shuffle and pseudorandom permutations. In: 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, pp. 1–6 (2016). https://doi.org/10.1109/iemcon.2016.7746312
2. Chowdhury, S.R., Hazra, T.K., Chakroborty, A.K.: Image encryption using pseudo random permutations. Am. J. Adv. Comput. **1**(1) (2014). http://dx.doi.org/10.15864/ajac.v1i1.2
3. Mallik, S., Saha, P., Singha Roy, A., Sinha, R., Hazra, T.K., Chakraborty, A.K.: Image hiding using zigzag and spiral traversal algorithms. Am. J. Adv. Comput. **1**(1) (2014)
4. Hazra, T.K., Chowdhury, S.R., Chakraborty, A.K.: Encrypted image retrieval system: a machine learning approach. In: 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, pp. 1–6 (2016). https://doi.org/10.1109/iemcon.2016.7746351
5. Hazra, T.K., Anand, A., Shyam, A., Chakraborty, A.K.: A new approach to compressed image steganography using wavelet transform. IOSR J. Comput. Eng. **17**(5), 53–59 (2015)
6. Hazra, T.K., Samanta, R., Mukherjee, N., Chakraborty, A.K.: Hybrid image encryption and steganography using SCAN pattern for secure communication. In: 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), Bangkok, pp. 370–380 (2017). https://doi.org/10.1109/iemecon.2017.8079625
7. Hazra, T.K., Ghosh, R., Kumar, S., Dutta, S., Chakraborty, A.K.: File encryption using Fisher-Yates shuffle. In: 2015 International Conference and Workshop on Computing and Communication (IEMCON), Vancouver, BC, pp. 1–7 (2015). https://doi.org/10.1109/iemcon.2015.7344521
8. Hazra, T.K., Mahato, A., Mandal, A., Chakraborty, A.K.: A hybrid cryptosystem of image and text files using blowfish and Diffie-Hellman techniques. In: 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), Bangkok, pp. 137–141 (2017). https://doi.org/10.1109/iemecon.2017.8079577
9. Li, C.: Cracking a hierarchical chaotic image encryption algorithm based on permutation. Sig. Process. **118**, 203–210 (2016)
10. Arroyo, D., Li, C., Li, S., Alvarez, G., Halang, W.A.: Cryptanalysis of an image encryption scheme based on a new total shuffling algorithm. Chaos, Solitons Fractals **41**(5), 2613–2616 (2009)
11. Pisarchik, A.N., Zanin, M.: Image encryption with chaotically coupled chaotic maps. Phys. D **237**(20), 2638–2648 (2008)
12. Chen, G.R., Mao, Y., Chui, C.K.: A symmetric image encryption scheme based on 3D chaotic cat maps. Chaos, Solitons Fractals **21**, 749–761 (2003)
13. Kocarev, L.: Chaos-based cryptography: a brief overview. IEEE Circ. Syst. Mag. **1**(3), 6–21 (2001)

14. Li, S.J., Zheng, X., Mou, X.Q., Cai, Y.L.: Chaotic encryption scheme for real-time digital video. Real-Time Imag. **VI**, 149–160 (2002)
15. Shujun, L., Xuanqin, M., Yuanlong, C.: Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography. In: Progress in Cryptology—INDOCRYPT 2001, pp. 316–329 (2001)
16. Ye, R.: A novel chaos-based image encryption scheme with an efficient permutation-diffusion mechanism. Optics Commun. **284**(22), 5290–5298 (2011)
17. Jolfaei, A., Mirghadri, A.: Image encryption using chaos and block cipher. Comput. Inf. Sci. **4**(1), 172 (2010)
18. Hazra, T.K., Kumari, N., Monica, Priya, S., Chakraborty, A.K.: Image encryption and decryption using phase mask over sinusoidal single and cross grating. In: 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, pp. 1–6 (2016). https://doi.org/10.1109/iemcon.2016.7746317
19. Mustafi, K., Sheikh, N., Hazra, T.K., Mazumder, M., Bhattacharya, I., Chakraborty, A.K.: A novel approach to enhance the security dimension of RSA algorithm using bijective function. In: 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, pp. 1–6 (2016). https://doi.org/10.1109/iemcon.2016.7746304
20. Sheikh, N.U., Hazra, T.K., Rahman, H., Mustafi, K., Chakraborty, A.K.: Multi-variable bijective mapping for secure encryption and decryption. In: 2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON), Bangkok, pp. 338–345 (2017). https://doi.org/10.1109/iemecon.2017.8079619