Original Article
ISSN (Online): 2456-8651

International Journal of Engineering Science Technologies
July-Aug 2022 6(4), 36-52

DIGITAL IMAGE ENCRYPTION USING RSA AND LFSR

Gandhi Srushti [1], Ravi Gor [2]

[1] Research Scholar, Department of Mathematics, Gujarat University, Gujarat, India
[2] Department of Mathematics, Gujarat University, Gujarat, India

Received 01 June 2022
Accepted 08 July 2022
Published 23 July 2022

CorrespondingAuthor
Gandhi Srushti,
srushtigandhi@gujaratuniversity.ac.in

DOI 10.29121/IJOEST.v6.i4.2022.351

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ABSTRACT

In this world of hasty evolution of exchanging digital data, data protection is essential to keep the data safe from the unauthorized parities. With the broad use of digital images of various fields, it is important to preserve the confidentiality for the data of an image from any unauthorized access. In this paper, the keys are generated using a random number generator which is grounded on Linear Feedback Shift Register (LFSR). The encryption/decryption is based on Rivest–Shamir–Adleman (RSA) with the random key generator.

Keywords: LFSR, RSA, Image Encryption-Decryption

1. INTRODUCTION

As digital images are vital in multimedia technologies, protecting user privacy is more critical. Encrypting the image to prevent against unauthorised access is critical to ensure the user's security and privacy. Several industries, including Internet communication, multimedia systems, diagnostic imaging, telemedicine, and military communications, adopt image and video encryption. The Internet and wireless networks, which take advantage of the rapid growth of multimedia and network technologies, are used to send, and store massive amounts of images. Since 1949, cryptography has played a vital role in security. This has been the battleground for mathematicians and scientists. AES, DES, RSA, IDEA, and some cryptographic methods are now implemented.

The image is most widely used by the means of communication in a variety of fields, including medicine, research, industry, and the military. The crucial image transfer will take place across an unauthorized Internet network. As a result, sufficient security is required to ensure that the image prohibits unwanted access to critical information. The image has the advantage of covering more multimedia information and hence requires security. Cryptography is a form of image security method that allows to send and save images securely over the Internet. Any system's prime priority is maintaining the integrity, secrecy, and authenticity of an image. Although cryptography is the most efficient method, it also has security issues when dealing with data through grey levels.

2. LITERATURE REVIEW

Anandakumar (2015) proposed an image encryption algorithm that worked efficiently. It is extremely secure, using little computational power and having a strong security. The simulation results suggested that the method had advantages based on their image-processing approaches. As a result, the algorithms are found to be effective for image encryption. It can provide security in open networks.

Kapur et al. (2015) implied a modest and secure procedure to protect images. The image encryption technique used two Pseudo Random Number generators. In the first step, the rows of the original image were switched using the Linear Feedback Shift Register technique. This was followed by the exchange of the columns. An intermediate cypher image was created as a result. The next stage involved replacing the intensity of each pixel in the intermediary cypher image by using Blum Shub algorithm. This produced the ultimate encrypted image.

Mondal et al. (2016) presented extremely secure encryption algorithm. They used permutation-substitution architecture for encryption and decryption of an image. Using Linear Feedback Shift Register, the image pixels of the plain image are scrambled during the permutation phase (LFSR). The output of this phase is an intermediary cipher image. This measures the same as the plain image in size. In the substitution period, sequence of random numbers was produced applying RC4 key stream generator. It was XORed with the pixel value of the intermediary cipher image to generate ultimate encrypted image. The experimental outcomes and security analysis of planned scheme was efficient and secure.

Chepuri (2017) projected RSA algorithm for encrypting images. The RSA algorithm has been updated to work with RGB images. The results of the experiments showed that they were able to successfully encrypt and decrypt a variety of images. This technique has a decent encryption effect. When compared to the original image, the cipher image created by their technique was completely different. This method offers enhanced security and is appropriate to secure image transmission over the Internet.

Jumaa (2018) solved the problem of secret key exchanging with the communicated parities by using a random number generator established by Linear Feedback Shift Register (LFSR). Random key generator was used to encrypt and decrypt the data using Advance Encryption Standard (AES). They also encrypted and decrypted grayscale and colour RGB images. Three elements were important to the functionality of the proposed system in their paper. The first dealt with the challenge of creating a secure encryption key that was random, the second with encrypting the plain or secret image using the AES technique, and the third with recovering the original image by decrypting the encrypted or cipher one.

Devi et al. (2018) offered unique medical image encryption procedure. For image confusion, they applied a Henon map, and for diffusion, they used a Linear Feedback Shift Register (LFSR). Researchers looked at the measured values and claimed that their system could withstand differential attacks. They show that their approach can ensure the security of DICOM images.

Jain and Sharma (2019) presented a technique for digital image encryption which is enhancing its security by using the RSA Algorithm.

3. TERMINOLOGIES

3.1. RSA ALGORITHM ANANDAKUMAR (2015)

The most widely used asymmetric digital image encryption algorithm is RSA. The oldest known algorithm for combining, authorizing and encryption, RSA (called for Rivets, Shamir, and Adelman, who initially presented it publicly), was one of the first important advancements in public-key cryptography. It employs a pair of keys, one of which is used for digital image encryption in such a way that only the other key in pair can authenticate it.

The keys are generated using a common technique, but it is not possible to generate them in a feasible way among them. The safety of RSA is built on the idea that factorize a larger number is complicated. It relies only on finding the prime factors that are applied in the process of encrypting and decrypting the digital image.

RSA algorithm for key generation is shown below:

With suitably long passwords, it is usually assumed that RSA is safe. Find two prime numbers and use those two prime numbers to generate a pair of keys.

Step 1: Firstly, select two distinct prime numbers p and q. Here, p and q should be the similar bit-length. Primality testing is a useful method for identifying prime integers.

Step 2: $n = pq$ calculate the value of n. The modulus for both the public and the private keys is n. The length of the key is often indicated in bits.

Step 3: Calculate $\varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1) = n - (p+q-1)$, where φ is Euler's totient function. This value is confidential.

Step 4: Select an integer e such that $1 < e < \varphi(n)$ and $gcd\ (e, \varphi(n)) = 1$; i.e., e and $\varphi(n)$ are co-prime. e is referred as the public key. e has a short bit-length and slight Mugging weight. This is more effective for encryption. However, it has been demonstrated that in some circumstances, lower values of e are less secure.

Step 5: Determine d as $d \equiv e - 1\ (mod\ \varphi(n))$; i.e., d is the modular multiplicative inverse of e $(modulo\ \varphi(n))$.

i.e., solve the d given $d \cdot e \equiv 1\ (mod\ \varphi(n))$. This is calculated using extended Euclidean algorithm. It uses pseudo code in the Modular integers section, inputs a and n are related to and $\varphi(n)$, respectively.

Step 6: Value of d value referred as the private key. The private key consists of the modulus n and the public key e.

The private key has the modulus n along with the private key d. p, q, and $\varphi(n)$ values are kept secret since they are used to calculate d.

The generated sequence is later converted into a sequence of 8-bit binary. It is used in the generating key sequences.

3.2. LINEAR FEEDBACK SHIFT REGISTER (LFSR) DEVI (2018)

A shift register called an LFSR has its input a liner function of its previous state. LFSR is built from simple shift register with a small number of XOR gates. Shift register is a form of digital circuit utilizing a flow of flipflops where the output of 1 flipflop is coupled with the input of the next.

Initial value of LFSR is known as seed. LFSR comprises of clocked storage elements (flipflops) and a feedback path. The amount of storage components determines the degree of LFSR. LFSR with m flipflops is said to be of m degree. As operation of register is deterministic, the stream of values generated by register is identified by its existing state. As register holds finite number, it must go through a repeating cycle. LFSR with relevant feedback function can produce a sequence of bits. It appears random and has exceptionally long cycle. LFSRs are n bits counter revealing pseudorandom comportment.

An m-stage linear feedback shift register (LFSR) is categorized by feedback polynomial of degree-m over GF (2). If the feedback polynomial is primitive, the sequence generated is periodic with period $(2^m - 1)$. Here, $m = 8$. There are 255 possible initial states. Each initial state generates a periodical sequence of the period $2^8 - 1 = 255$. The sequence produced with different initial states are shifted versions of one another.

Figure 1

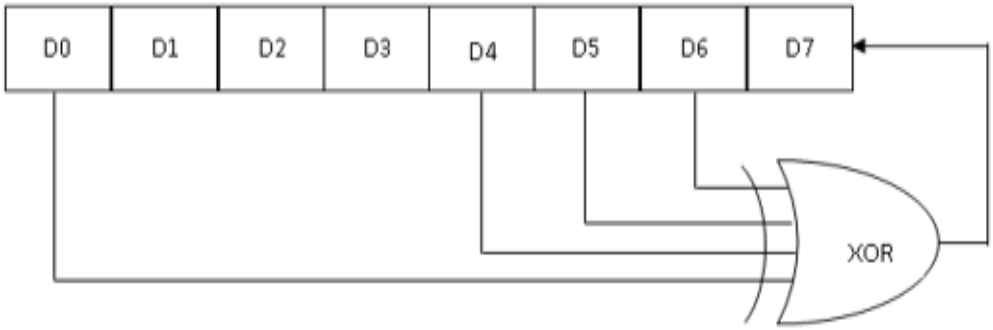

Figure 1 Linear Feedback Shift Register

LFSRs are applied in numerous key stream generators for the reason:

- LFSRs are good for hardware implementation.
- They can generate sequences with good statistical properties.
- They can generate sequences of large period.
- They may be quickly studied using algebraic methods due to their composition.

In the purposed scheme, sequences of 8 bit are used for generating key sequences. The sequence is denoted as $\{K_2\}$.

4. PROPOSED WORK

4.1. ENCRYPTION ALGORITHM

Step 1: An 8-bit image of size $M \times N$ pixels is converted into a one-dimensional array of pixel $P_i = \{P_1, P_2, \ldots, P_n\}$, where $i = 1,2,3 \ldots . n$ and $n = M \times N$. Then, each unsigned pixel value between 0 and 255 is converted into an 8-bit block.

Step 2: Bit by bit XOR operation is done between sequence $\{K_1\}$ generated by RSA and $\{K_2\}$ produced from LFSR using $\{K_1\}$ to obtain final key sequence $\{K_i\}$.

$$\mathrm{K_i} = \mathrm{K_1} \oplus \mathrm{K_2}$$

Step 3: The binary image pixels $P_i$are XORed with key sequence $\mathrm{K_i}$ to get encrypted pixel $\{C_i\}$. This block of 8-bit subsequently converted into decimal digit $\{C'_i\} \in 0$ to 255.

$$\mathrm{C_i} = \mathrm{P_i} \oplus \mathrm{K_i}$$

Step 4: Repeat step 3 to encrypt the entire image pixels. Convert all encrypted digits $[\![C'_i = \{C'_1, C'_2, \ldots, C'_n\}$ into an array of size $M \times N$ to obtain the encrypted image.

4.2. DECRYPTION ALGORITHM

Step 1: Encrypted image of size $M \times N$ pixels is transmuted into one dimensional array of pixels$C'_i = \{C'_1, C'_2, \ldots, C'_n\}$ where $i = 1, 2, 3$ and $n = M \times N$. Later, convert each unsigned pixel values into a block of 8 bit.

Step 2: The key sequence $\{K_i\}$ is attained by $\{K_1\}$ and $\{K_2\}$ as defined in Step:3 of encryption algorithm. It is used to decrypt image. The obtained block of decrypted 8-bit is converted into decimal digits called $\{D'_i\}$.

$$\mathrm{D'_i} = \mathrm{C'_i} \oplus \mathrm{K_i}$$

Step 3: One dimensional array of decrypted pixels $D_i = \{D_1, D_2, \ldots, D_n\}$ is the transformed into an array of size $M \times N$ to get decrypted image.

5. EXAMPLE

Figure 2

Figure 2 Original Image Pinterest (2021)

Using PYTHON.

Image array: <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=368x258 at 0x122523A2FA0>

Numpy array: <class 'numpy.ndarray'>

Image shape (258, 368, 3)

Pillow image: <class 'PIL.Image.Image'>

Image mode: RGB

Image size: (368, 258)

Image array/in matrix form:

```
[[[36  165  170                                  [0  89  107
  36  165  170          ..............            1  90  108
  36  166  168]                                   4  93  111]]

 [[37  166  171                                  [0  89  107
  37  167  169          ..............            1  90  108
  38  168  170]                                   4  93  111]]

 [[37  167  169                                  [0  89  107
  37  167  169          ..............            1  90  108
  38  168  170]                                   4  93  111]]

    ..............      ..............              ..............
    ..............      ..............              ..............

 [[83  173  169                                  [150  150  116
  90  173  169          ..............           149  150  118
  90  160  170]                                  159  162  131]]

 [[91  176   21                                 [[111  113   92
  101  180   39          ..............         143  142  121
  85  155   30]                                 180  178  155]]

[[111  196   41                                 [115  117   96
 119  198   57          ..............         131  130  109
 102  172   47]                                 168  166  143]]
```

Total pixel value: 284832

Key generation using RSA:

Here, $p = 11$ and $q = 23$ is taken for this example. Any large primes can also be chosen to get better result. Using this, the key

obtained is 210 which is converted into binary 11010010.

Key generation using LFSR:

Using RSA key 11010010 and a primitive polynomial $x^8 + x^7 + x^5 + x^3 + 1 = 0$ for 8 bits; the output is.

Table 1

Table 1					
11010010	0	00011100	1	11010000	0
01101001	1	10001110	0	01101000	0
10110100	1	01000111	0	00110100	1
11011010	0	00100011	1	10011010	0
01101101	1	10010001	1	01001101	0
10110110	0	11001000	1	00100110	0
01011011	1	11100100	1	00010011	0
10101101	1	11110010	0	00001001	0
11010110	1	01111001	1	00000100	0
11101011	0	10111100	0	00000010	1
01110101	0	01011110	0	10000001	1
00111010	1	00101111	0	11000000	0
10011101	0	00010111	0	01100000	1
01001110	0	00001011	1	10110000	1
00100111	1	10000101	1	11011000	1
10010011	0	11000010	1	11101100	0
01001001	0	11100001	0	01110110	0
00100100	1	01110000	1	00111011	0
10010010	1	10111000	0	00011101	0
11001001	0	01011100	1	00001110	0
01100100	1	10101110	1	00000111	0
10110010	0	11010111	0	00000011	0
01011001	0	01101011	0	00000001	1
00101100	1	00110101	0	10000000	0
10001011	1	00011010	0	01000000	0
11000101	1	00001101	0	00100000	1
11100010	0	00000110	1	10010000	0
01110001	0	10000011	0	01001000	1
00111000	0	01000001	1	10100100	1
		10100000	1	11010010	(Repeating)

89th clock, period will be repeating and will give pseudorandom sequence.

So, 10100100 is considered as key K_2.

5.1. ENCRYPTION ALGORITHM

Step 1: An 8-bit image of size $M \times N$ pixels is converted into a one-dimensional array of pixel $P_i = \{P_1, P_2, \ldots, P_n\}$, where $i = 1,2,3 \ldots . n$ and $n = M \times N$. Then, each unsigned pixel value between 0 and 255 is converted into an 8-bit block.

A one-dimensional array of pixel $P = \{P_1, P_2, \ldots, P_n\}$:

[[36 165 170 … 4 93 111]

[37 166 171 … 4 93 111]

[37 167 169 … 4 93 111]

…

[83 173 23 … 159 162 131]

[91 176 21 … 180 178 155]

[111 196 41 … 168 166 143]]

Step 2: Bit by bit XOR operation is employed between sequence $\{K_1\}$ which is generated by RSA and $\{K_2\}$ which is generated by LFSR using by $\{K_1\}$ to obtain final key sequence $\{K_i\}$.

$$K_i = K_1 \oplus K_2$$

$$K_1 = 11010010$$

$$K_2 = 10100100$$

$$K_1 \oplus K_2 = 01110110$$

Input a binary number: 01110110.

The decimal value of the number is 118.

Step 3: The binary image pixels P_i are XORed with key sequence $\{K_i\}$ to get encrypted pixel $\{C_i\}$. This block of 8-bit subsequently converted into decimal digit $\{C'_i\} \in 0$ to 255.

$$C_i = P_i \oplus K_i$$

$array([[82, 211, 220, \ldots, 114, 43, 25]$

$[83, 208, 221, \ldots, 114, 43, 25]$

$[83, 209, 223, \ldots, 114, 43, 25]$

…,

$[37, 219, 97, \ldots, 233, 212, 245]$

$[45, 198, 99, \ldots, 194, 196, 237]$

$[25, 178, 95, \ldots, 222, 208, 249]], dtype = uint8)$

Step 4: Repeat step 3 to encrypt the entire image pixels. Convert all encrypted digits $C'_i = \{C'_1, C'_2, \ldots, C'_n\}$ into an array of size $M \times N$ to obtain the encrypted image.

Data type: Unit 8

Figure 3

Figure 3 Encrypted image

For encrypted image:

Total pixel value: 854496

Image array:$< PIL.JpegImagePlugin.JpegImageFile\ image\ mode = L\ size = 1104x258\ at\ 0x1C3A3AAB070 >$

NumPy array: <class 'numpy. ndarray'>

Image shape: (258,1104)

5.2. DECRYPTION ALGORITHM

Step 1: Encrypted image of size $M \times N$ pixels is transmuted into one dimensional array of pixels $[\![C'_i = \{C'_1, C'_2, \ldots, C'_n\}$, where $i = 1, 2, 3$ and $n = M \times N$. Later, convert each unsigned pixel values into a block of 8 bit.

$array([[\ 82, 211, 220, \ldots, 114,\ 43,\ 25],$

$[\ 83, 208, 221, \ldots, 114,\ 43,\ 25],$

$[\ 83, 209, 223, \ldots, 114,\ 43,\ 25],$

$\ldots,$

$[\ 37, 219,\ 97, \ldots, 233, 212, 245],$

$[\ 45, 198,\ 99, \ldots, 194, 196, 237],$

$[\ 25, 178,\ 95, \ldots, 222, 208, 249]], dtype = uint8)$

Step 2: The key sequence $\{K_i\}$ is attained by $\{K_1\}$ and $\{K_2\}$ as defined in Step:3 of encryption algorithm. It is used to decrypt image. The obtained block of decrypted 8-bit is converted into decimal digits called $\{D'_i\}$.

$$D'_i = C'_i \oplus K_i$$

$$D'_i = C'_i \oplus K_i$$

$array$ [[36 165 170 … 4 93 111]

[37 166 171 … 4 93 111]

[37 167 169 … 4 93 111]

…

[83 173 23 … 159 162 131]

[91 176 21 … 180 178 155]

[111 196 41 … 168 166 143]]

Figure 4

Figure 4 D'_i image

For D'_i image.

Total pixel value: 854496

Image array: < PIL. PngImagePlugin. PngImageFile image mode = L size = 1104x258 at 0x1C3A3A70610 >

Numpy array: <class 'numpy. ndarray'>

Image shape: (258,1104)

Step 3: One dimensional array of decrypted pixels $D_i = \{D_1, D_2, \ldots, D_n\}$ is the transformed into an array of size $M \times N$ to get decrypted image.

```
[[[36  165  170                                  [0   89   107
  36  165  170          ... ...                   1   90   108
  36  166  168]                                   4   93   111]]
 [[37  166  171                                  [0   89   107
  37  167  169          ..........                1   90   108
  38  168  170]                                   4   93   111]]
 [[37  167  169                                  [0   89   107
  37  167  169          ..........                1   90   108
  38  168  170]                                   4   93   111]]
    ..........          ..........                  ..........
    ..........          ..........                  ..........
 [[83  173  169                                  [150  150  116
  90  173  169          ..........               149  150  118
  90  160  170]                                  159  162  131]]
 [[91  176   21                                  [[111  113   92
  101  180   39          ..........              143  142  121
  85  155   30]                                  180  178  155]]
[[111  196   41                                  [115  117   96
  119  198   57          ..........              131  130  109
  102  172   47]                                 168  166  143]]
```

Figure 5

Figure 5 Decrypted image

6. RESULTS AND DISCUSSION

6.1. VISUAL TESTING

The visual testing is done online on https://www.textcompare.org/image/.

Figure 6

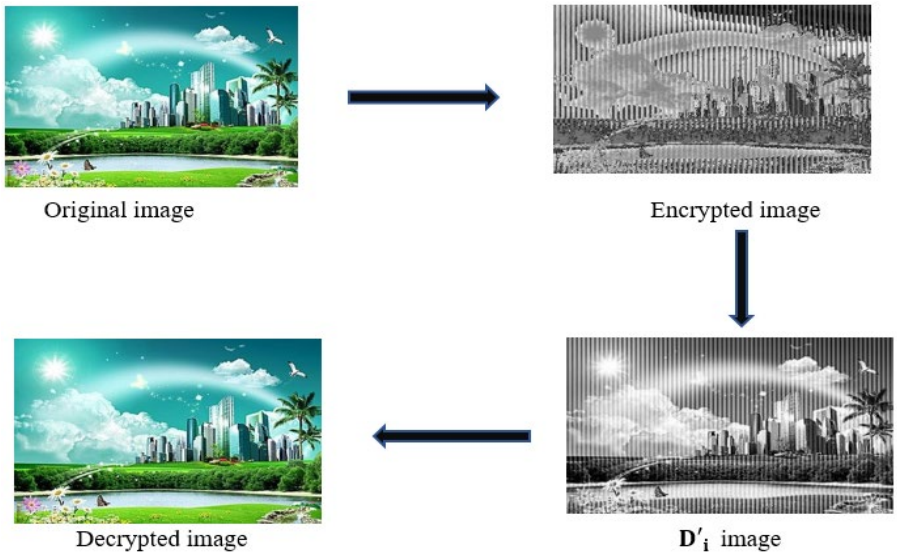

Figure 6 Visual Testing

Comparing original and encrypted image, similarity between them is shown in the below table. White dotes show the similar pixel values of original image and encrypted image.

By comparing original image with encrypted image and D'_i image, the difference is shown in below table. The white dot indicates the similar pixel value of original image and encrypted image. The difference is found with full transparency when less ignored along with original size and movement with different intensity.

Figure 7

Figure 7 Difference between original and D'_i image

Figure 8

Figure 8 Difference between original and encrypted image

Table 2

Table 2

Image	Difference
Encrypted image	99.88%
D'_i image	99.55%

By comparing original image with encrypted image and D'_i image, the difference is shown in below table. The white dot indicates the similar pixel value of original image and encrypted image. The difference is found with full transparency when less ignored along with scale to same size and movement with different intensity.

Figure 9

Figure 9 Difference between original and D'_i image

Figure 10

Figure 10 Difference between original and encrypted image

Table 3

Table 3

Image	Difference
Encrypted image	99.78%
D'_i image	99.97%

6.2. SENSITIVITY ANALYSIS

Image quality and vision outcomes were generated by the experiment. The following parameters are used to evaluate image quality:

6.2.1. NUMBER OF PIXELS CHANGE RATE (NPCR)

When the difference between two encrypted images is negligible, NPCRs are used to verify the number of changing pixels among themselves. The NPCR can be mathematically defined as follows:

$$\text{NPCR} = \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{D(i,j)}{M \times N} \ 100\%$$

$$\text{Where } D(i,j) = \begin{cases} 0; & if\ C_1(i,j) = C_2(i,j) \\ 1; & if\ C_1(i,j) \neq C_2(i,j) \end{cases}$$

m, n is the weight and height of the encrypted interferogram,

$C_1(i,j)$ is the interferogram encrypted before pixel change,

$C_2(i,j)$ is the interferogram encrypted after pixel change,

$D(i,j)$ is the bipolar network.

The optimal NPCR value is:

Table 4

Table 4

Image	NPCR
Encrypted image	100%
D'_i image	100%

This experimental result is executed in python

6.2.2. MEAN SQUARED ERROR (MSE) AND PEAK SIGNAL TO NOISE RATIO (PSNR)

The peak signal-to-noise ratio (PSNR) between any two images is examined in decibels by PSNR block investigates. The PSNR ratio compares the quality of the original and encrypted images. The PSNR increases with the quality of the compressed or rebuilt image.

To compare image compression quality, the mean square error (MSE) and peak signal-to-noise ratio (PSNR) are evaluated. The MSE is a rate of the peak error between the encrypted and original image, whereas the PSNR is a measure of the cumulative squared error.

The smaller the MSE value, the smaller the error.

The PSNR is obtained by first calculating the mean-squared error (MSE) utilizing the equation:

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - k(i,j)]^2$$

$$= \sum_{i,j} \frac{[I(i,j) - k(i,j)]^2}{mn}$$

PSNR can be calculated as:

$$\text{PSNR} = 20\ \log_{10} \frac{256}{\sqrt{MSE}}$$

$$= 20 \log_{10} I(i,j) - 10 \log_{10} MSE$$

The optimal MSE and PSNE values are:

Table 5

Table 5

Image	MSE	PSNR
Encrypted image	1139328	12.40169 *db*
D'_i image	1139328	12.40169 *db*

This experimental result is executed in python

6.2.3. UNIFIED AVERAGE CHANGING INTENSITY (UACI)

UACI is used to calculate the average intensity of the difference between two encrypted images (C_1 and C_2). It is used to determine the strength of an encryption scheme. Its quality is determined by the format and size of the image. The average intensity variation between the ciphered and original images is measured using UACI. The highest UACI suggests that the recommended technique is resistant enough for certain attacks.

For an image of size m× n, UACI is calculated as follows:

$$\text{UACI} = \frac{1}{mn} \sum_{i,j} \frac{|C_1(i,j) - C_2(i,j)|}{256}\ 100\%$$

From the encrypted image and D'_i image, the optimum UACI value is 0%.

On comparing the original image with encrypted image and D'_i image, the optimum UACI value is 78.125%.

6.2.4. ENTROPY ANALYSIS

A random event's level of uncertainty is measured by information entropy. The event contains a significant amount of information. With uncertainty or unpredictability, it increases. It is used in a variety of applications, including encryption, lossless data compression, and statistical inference. The entropy $H(m)$ of m can be calculated as Kapur et al. (2015)

$$H(m) = -\sum_{i=0}^{255} P(x_i) \log_2 P(x_i)$$

Where L is the total number of symbols.

$m_i \epsilon\ m$ and $p(x_i)$ is the probability of symbol x_i.

For the original digital image, $H(m)$ should theoretically be equal to 8 as there are 256 values of the data source in red, blue, and green colours of the image with the same probability.

Entropy values are:

Table 6

Table 6

Image	Entropy
Original image	7.35
Encrypted image	7.98
D'_j image	7.98

6.2.5. TIME TAKEN FOR ENCRYPTION AND DECRYPTION OF AN IMAGE

The time taken to encrypt the image is 0.0 sec and time taken to decrypt the image is 3.796875 sec.

7. CONCLUSION

The key utilized for image encryption and decryption in this paper is produced using RSA and Linear Feedback Shift Register (LFSR). The suggested method is extremely sensitive to the LFSR's initial state. An RSA generates the first key, and LFSR uses the first key to generate the second key. Then both keys are XORed together to produce a strong key, which is the final key. As a result, in terms of guessing that key, the hacker will have a great difficulty. Consequently, if the wrong key is used, the image will be radically different.

The results show original and encrypted image is extremely uncorrelated and unique. RSA uses more controlled parameters as compared to another algorithms, which enhances the data security. The computations and coding were done using the capabilities of PYTHON.

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Anandakumar, S. (2015). Image cryptography using RSA algorithm in network security. International Journal of Computer Science & Engineering Technology, 5(9), 326-330.

Chepuri, S. (2017). An RGB image encryption using RSA algorithm. International Journal of Current Trends in Engineering & Research (IJCTER), 3(3), 1-7.

Devi, S. R. Rajarajeswari, V. Thenmozhi, K. RengarajanAmirtharajan, and Praveenkumar, P. (2018). Henon and LFSR assisted key based encryption. International Journal of Pure and Applied Mathematics, 119(16), 455-460.

Jain, A. & Sharma, S. (2019). A Novel Digital Image Encryption Method Based on RSA Algorithm. 11(1), 650-654.

Jumaa, N. K. (2018). Digital image encryption using AES and random number generator. Iraqi Journal of Electrical and Electronic Engineering, 14(1). https://doi.org/10.37917/ijeee.14.1.8

Kapur, V. Paladi, S. T. & Dubbakula, N. (2015). Two level image encryption using pseudo random number generators. International Journal of Computer Applications, 115(12). https://doi.org/10.5120/20200-2446

Pinterest (2021).

TextCompare (2021).

Mondal, B. Sinha, N. & Mandal, T. (2016). A secure image encryption algorithm using lfsr and rc4 key stream generator. In Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics, 227-237. https://doi.org/10.1007/978-81-322-2538-6_24