Routledge
Taylor & Francis Group

Journal of Applied Security Research

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/wasr20

Enhancing Multimedia Security Using Shortest Weight First Algorithm and Symmetric Cryptography

Karishni Mehta, Glen Dhingra & Ramchandra Mangrulkar

To cite this article: Karishni Mehta, Glen Dhingra & Ramchandra Mangrulkar (2022): Enhancing Multimedia Security Using Shortest Weight First Algorithm and Symmetric Cryptography, Journal of Applied Security Research, DOI: 10.1080/19361610.2022.2157193

To link to this article: https://doi.org/10.1080/19361610.2022.2157193

Published online: 19 Dec 2022.

JOURNAL OF APPLIED SECURITY RESEARCH
https://doi.org/10.1080/19361610.2022.2157193

Routledge
Taylor & Francis Group

Enhancing Multimedia Security Using Shortest Weight First Algorithm and Symmetric Cryptography

Karishni Mehta, Glen Dhingra, and Ramchandra Mangrulkar

Computer Engineering, Dwarkadas J. Sanghvi College of Engineering, Mumbai, India

ABSTRACT

A critical component of information security is data encryption, and numerous researchers have invented various approaches to encrypt data. This paper presents a novel image encryption approach using the shortest weight first algorithm from graph theory, categorized as a symmetric cipher where the user has much freedom in selecting the keys. Furthermore, this method requires no sophisticated computations and is simple to understand and implement while providing effective protection against cryptographic attacks, such as brute force. The detailed algorithm, analysis, and security implications are also presented. The results firmly set a good base for many real-world applications involving security.

KEYWORDS

Shortest weight first; graph theory; cryptography; cryptography attacks; symmetric encryption

1. Introduction

In the twenty-first century, technology is one of the, if not most essential aspects of human lives. A sizable portion of that includes communication infrastructure. Unfortunately, the information transmitted between devices is often sensitive and only meant for a particular device or user. To ensure the data does not fall into the wrong hands, there is a field of computer science called Information Security. Information security prevents unauthorized alteration, access, delay, disclosure, usage, signal, inspection, or destruction of data when transmitted through different communication channels. One of the most formidable techniques to keep our data and information safe between peers is to use encryption. It renders data impenetrable and unreadable, rendering it useless even if it falls into the wrong hands (Sakthidasan & Santhosh Krishna, 2011).

The financial institutions, government, military, and hospitals all deal with sensitive information, including photos of patients, financial conditions, geographic locations, and enemy positions. Most of this data is collected and stored on computers and then communicated via the internet

CONTACT Ramchandra Mangrulkar ramchandra.mangrulkar@djsce.ac.in Dwarkadas J. Sanghvi College of Engineering, Mumbai, Maharashtra, India.

(Anandakumar, 2015). As a result, government and private enterprises and individuals must be provided with security for multimedia information (Ak et al., 2014). Multimedia cryptography can make it difficult for unauthorized users to decrypt the data by transforming it into a completely different format (Hu et al., 2015; Storms, 2016).

Many symmetric and asymmetric algorithms, such as AES, RSA, and triple DES, are used today. In addition, numerous research has contributed to various cryptographic techniques, each with its own principles, efficiencies, working scopes, and application domains. In (Gao et al., 2020), the author uses a unique key encapsulation technique and a data encapsulation technique to illustrate a new hybrid asymmetric encryption system. To authenticate data propagation and individually secure group communication (Hsieh et al., 2011), uses an asymmetric key strategy and group-based Elliptic Curve cryptography. Finally (Tahat & Abdallah, 2018), proposes an algorithm that uses chaotic maps and factorization problems and has a hybrid public verifiable authentication process (Siahaan, 2017). have used an encryption technique, R4C, derived from RSA, to alter the image's pixel intensities.

Many researchers have also used chaotic maps to encrypt images (Ahmad & Shamsher Alam, 2009); introduced a symmetric system based on three different chaotic maps, whereas (Chen et al., 2004) is also a symmetric system but uses a three-dimensional chaotic cat map. The author of (Pourasad et al., 2021) uses a fast and secure technique of encrypting images using random chaos sequences. In (Kamal et al., 2021), using zigzag patterns, permutations, and rotation, an image is split and shuffled, and then a key is formed using a chaotic generating map, which diffuses the scrambled image.

Artificial neural networks have been used in cryptography to encrypt images. For example (Dey & Paul, 2019), uses a single-layer artificial neural network and a chaotic map to encrypt the image, whereas (Man et al., 2021) uses dynamic adaptive diffusion and convoluted neural networks for double-image encryption. Furthermore, components of DNA are used by (Nandy et al., 2021) to hide plain text from attackers; this approach uses the assignment of DNA codes to colors and a random key generation. Finally (Najjar & Saleh, 2017), implements image encryption by using the hex function to get HEX code and RSA (Rivest, Shamir, and Adleman) public key algorithm to generate the text of the cipher image.

In particular, numerous scholars are researching graph theory principles that can be used in various aspects of cryptography. For example (Mittenthal, 2007; Patanwadia & Mangrulkar, 2021), uses Hadamard encoding by generating random Hadamard matrices from a strongly regular graph. In contrast, in (Abduljaleel et al., 2021), a color image of a graph

theory-based encryption technique has been proposed. The key generation uses audio signals passed through various stages based on graph theory (Omotosho & Emuoyibofarhe, 2015). introduces a key generation method from an image where textural features were extracted from partitioned images to generate cryptographic keys.

Many academics and scientists utilize steganography to hide information (Gopalan & Kumaresan, 2020; Mehta et al., 2022; Mhatre et al., 2022); presents a method for reversibly hiding data in encrypted images, which uses cellular memory automata. In addition (Jeyaprakash et al., 2022), introduces a data hiding scheme that uses the RSA algorithm and a multidirectional PVD; whereas (Joseph et al., 2013) provides a novel reversible data concealing system based on a dynamic process to select ideal peak valley pairs to optimize the object's embedding capacity. Finally, a novel steganography method is proposed in (Jeppiaar, 2015), which uses reversible and histogram shifting techniques to conceal data.

The author presents a novel encryption algorithm for multimedia data, such as images, based on graph theory in this paper. The methodology is named as SWF (Shortest Weight First) algorithm.

The rest of the paper is organized as follows: Section 2 elaborates on the working of the SWF algorithm. Next, experimentation has been presented, and the results are given in Section 3. These results and a detailed analysis of the SWF algorithm and its security aspects are presented in Section 4. Finally, section 5 concludes the paper with possible future directions for further extension.

2. SWF algorithm

In this section, the author introduces and explains an inventive approach to multimedia cryptography.

2.1. Encryption

The original image to be encrypted using the SWF approach is taken in jpeg/jpg format and turned into a three-dimensional matrix with dimensions of (height, width, 3), where height and width are the image's dimensions. The RGB channels are represented by the number three, which is the third dimension. When supplied into the red, green, and blue inputs, an RGB image depicts three different layers of the image stacked on top of each other, producing a color image on the screen. Each color pixel has three values that correspond to the RGB image's red, green, and blue color components at a specific spatial location. The amount of red, green, and

blue stored in each color plane at the pixel's location determines the color of any pixel as a result.

The author's approach uses the shortest weight first algorithm and graph theory to encrypt the input multimedia data. First, a collection of visited nodes is formed in the shortest weight first algorithm to keep track of vertices that have been visited. A node, say *k*, is chosen from the start node whose edge originating from the start node has the lowest cost. The starting node is then added to the visited nodes collection so that the algorithm does not choose it again later in the process. The next node in the visited nodes set is selected from k, whose edge coming from k has the lowest cost. The algorithm traverses all the nodes in the graph until the visited nodes set contains them.

After taking the input image, the algorithm generates a fully connected, weighted, and directed graph of *n* nodes. A fully connected graph is where each node is connected to the other $n-1$ nodes. When every edge of a graph is assigned a cost, weight, or value, it is called a weighted graph. A directed graph is one in which each edge has a start node and an end node, represented by an arrow, with the end node being the node at the pointed edge.

Following the formation of the graph, the algorithm must assign a cost to each edge, which necessitates the creation of a cost matrix. This algorithm uses key1, key2, and key3, each of which contributes to the cost matrix generation. For a graph with n nodes, a cost matrix of dimension $n \times n$ is required. The algorithm does not build the complete $n \times n$ matrix at once; instead, it extracts values from the image first, then uses the keys to generate more values until the total number of generated values reaches *n*. This is how the algorithm retrieves the first row of the matrix. Using this initial row of n data and key3, the algorithm creates the additional $n-1$ rows, resulting in the $n \times n$ cost matrix.

As previously said, each image is converted into a three-dimensional shape matrix (height, width, 3); therefore, the total number of numerical values in this matrix is m = height*width*3, while each value's range is 0–256. Thus, the matrix has a total of m elements; however, many values are repeated because the range is limited. To create the cost matrix, sort all the unique values from the picture matrix in ascending order and take the first 50 values.

The 50 values extracted from the image form the first 50 values of the first row; the algorithm now uses key1 and key2 to get the rest of the $n-50$ values. Key1 and key2 are two large polynomial equations with two input variables, *x*, and *y*. Let the first two numbers from the 50 values be *p* and *q*, so input p and q into key1 as $x=p$ and $y=q$ to get one value, and again input the same numbers, *p*, and *q*, into key1 as $x=q$ and $y=p$

to get another value. Therefore every pair of numbers gives it two unique values. After putting numbers through the key1, the algorithm generates 150 values. The algorithm again inputs pairs of values into key2 as done with key1 to generate more values. If the total number of values obtained after key1 and key2 is greater than n, the first n values are considered.

Now the algorithm has n values; thus, it has the first row of the cost matrix. Next, it uses key3 to obtain the whole $n \times n$ matrix. Key3 is a large polynomial equation with only one variable, x. To get the second row of the matrix, it takes the first row and left shift it with a certain number; to get the third row of the matrix, it takes the first row and right shift it with a certain number, and to get the fourth row, it left shifts the first row. This is done till all n rows are calculated. To obtain the value, the algorithm needs to shift the row; it inputs the row number into key3 and gets an integer value to shift the row by. So to generalize, in order to get the kth row, the algorithm first checks if k is odd or even; if k is even, the first row is to be shifted to the left, and if k is odd, the first row is to be shifted to the right. To get the number by which the row is to be shifted, the algorithm inputs k into key3 to get an integer.

Repeating the above steps $n-1$ times, the algorithm gets all n rows of n values each, thus successfully generating the $n \times n$ cost matrix. The algorithm now has the $n \times n$ cost matrix and the fully connected graph with n nodes. It now divides the image matrix into $n-1$ horizontal parts; since one node of the graph is the starting node, it needs $n-1$ parts of the image matrix.

From here on, the algorithm starts encrypting the image; since it is a fully connected graph, the start node is connected to all the other nodes; it compares the cost of each edge, starting from the start node, and finds the node, g1, whose cost is the minimum. The algorithm initializes an empty set called visited_nodes, where it saves all the visited nodes. Thus the start node is added to this set. The algorithm also initializes an integer xor_pointer to the cost index value in the cost matrix, which keeps track of the number it has to perform the xor operation with.

The first part of the image is then xor-ed with the xor_pointer and stored in the node g1; from g1, the algorithm again selects the next node g2 with the lowest cost but does not consider visited nodes; in this case, the start node. Next, the xor_pointer is incremented by the index value of node g2 in the cost matrix, the second part of the image is then xor-ed with the xor_pointer and stored in g2, and g1 is added to the visited_nodes set. Following this, the algorithm traverses through the whole graph, choosing the shortest node, performing the xor operation, and saving a part of that image in that node. So now, the algorithm has a graph, with each node containing a part of the image. It now arranges these nodes in the

ascending order of position of nodes as given in the cost matrix. The last step is to perform a bit-wise not operation on every pixel in the new image formed, and the encrypted image is obtained. Figure 1 represents the block diagram of encryption, and Figure 2 shows the schematic diagram for encryption.

This algorithm follows the Shortest Weight First Approach. To represent it mathematically, the author assumes the current node to be U and the next node to be visited to be V. Set S contains indexes of all the unvisited nodes. Set C contains the weight of all nodes from the current Node U The algorithm looks for the node closest to Node U, which is yet to be visited in the graph.

$$V = \min\ C\colon \left(\min C_{\text{weights from U}} \in S_{\text{unvisited nodes}}\right) \tag{1}$$

2.2. Decryption

Generation of the kxk cost matrix is the same as done in encryption. First, the encrypted image is taken as input and converted into a three-dimensional matrix, as done for encryption. The image is then divided into *k* parts; every part is stored in its corresponding node. The algorithm initializes an empty set called visited_nodes, where it saves all the names of the visited nodes; thus, the start node is added to this set. The algorithm also initializes an integer xor_pointer to the index value of the minimum cost

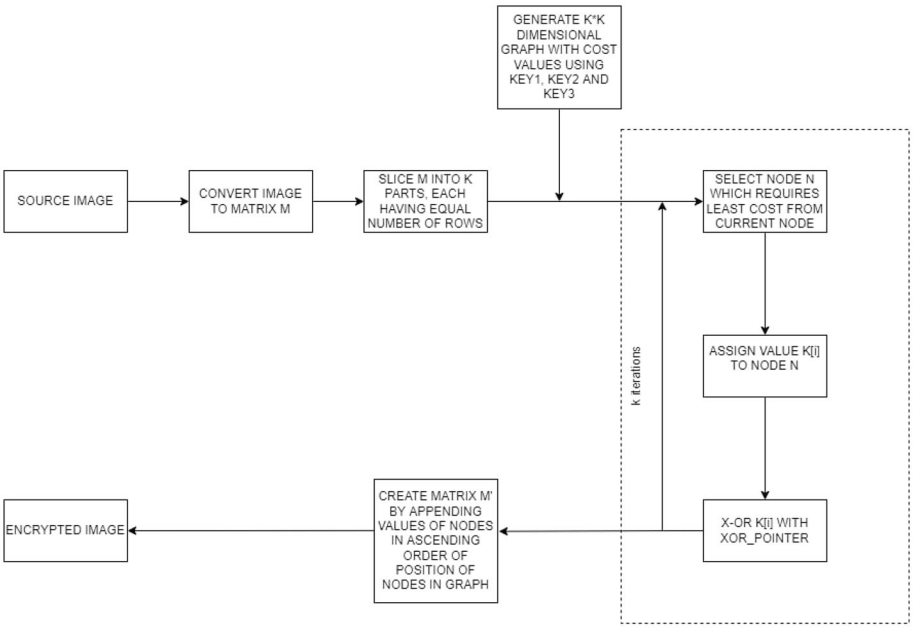


Figure 1. Block diagram of Shortest Weight First Encryption Algorithm.

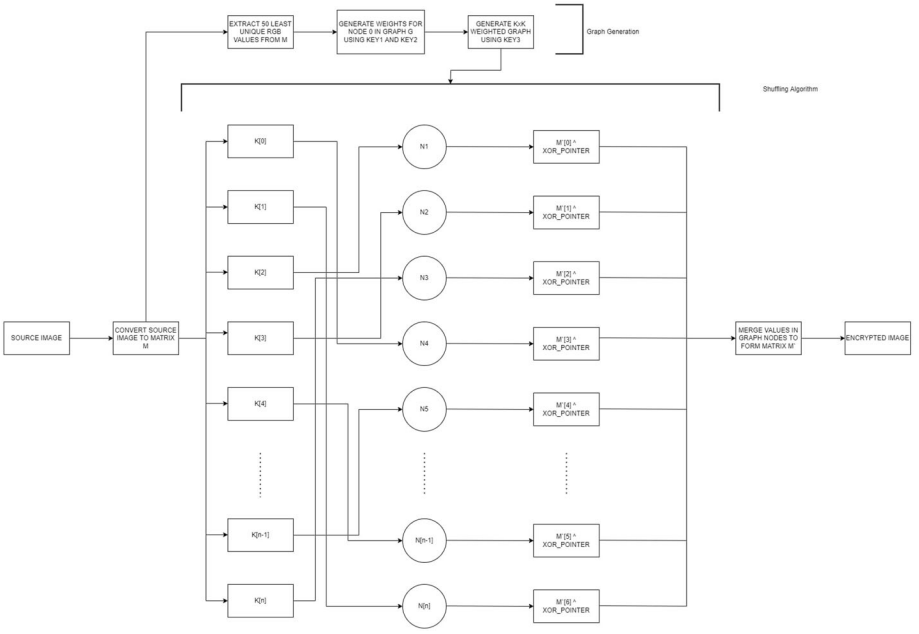


Figure 2. Schematic representation of Shortest Weight First Encryption Algorithm.

from the start node in the cost matrix, which keeps track of the integer with which it has to perform the xor operation. From the start node, the shortest weight node is selected, and the part of the image corresponding to that node is xor-ed with the xor pointer and is then appended to the decrypted matrix. The xor_pointer is incremented in the same way as in encryption. From that node, the next node with the least cost is selected, and that part of the image is xor-ed and appended to the matrix. In this way, all the nodes are visited, and their part of the image is xor-ed and added to the matrix. Figure 3 represents the block diagram of decryption process, and Figure 4 shows the schematic diagram for decryption.

2.3. Algorithm

2.3.1. Cost matrix generation

$img \leftarrow image_{input}$
$cost_{matrix} \leftarrow [\,]$
$unique_{arr} \leftarrow$ 50 unique values from image
$i \leftarrow 0$
while $i \leq size(unique)$ **do**
  $x \leftarrow unique[i]$
  $y \leftarrow unique[i+1]$
  $a \leftarrow key1(x, y)$
  $swap(x,\ y)$

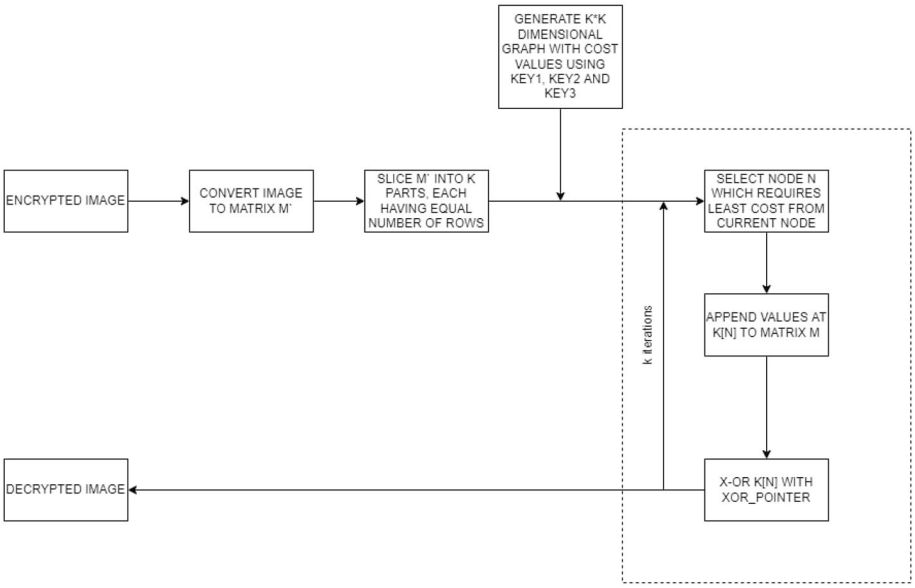


Figure 3. Block diagram of Shortest Weight First Decryption Algorithm.

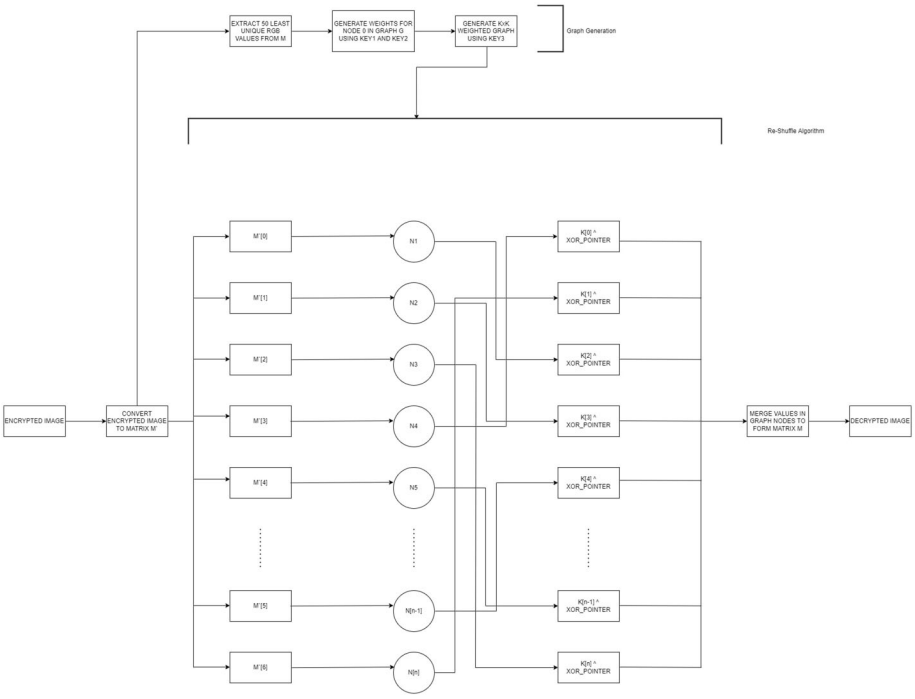


Figure 4. Schematic representation of Shortest Weight First Decryption Algorithm.

```
        unique.add(a)
        unique.add(b)
        i++
    end while
    i ← 0
    while i ≤ size(unique) do
        x ← unique[i]
        y ← unique[i + 1]
        a ← key2(x, y)
        swap(x, y)
        unique.add(a)
        unique.add(b)
        i++
    end while
    i ← 0
    while i ≤ size(unique) do
        if i%2 == 0 then
            left_shift(unique[i], key3[i])
        else
            right_shift(unique[i], key3[i])
    end if
    i++
end while
```

2.3.2. Encryption

```
    x ← [ ]
    for i ← 0; i ≤ matrix.shape[0]; i++ do
        x.append([])
    end for
    block_size ← img.shape[0]/cost_matrix.shape[0]
    block_start ← 0
    block_end ← block_size
    xor_pointer ← 1
    visited_nodes ← []
    i ← 0
    while size(visited_nodes)! = size(x) do
        arr ← matrix[minimum_index]
        min ← 0
        for i ← 0; i ≤ size(arr); i++ do
            if i! = minimum_index and arr[i] < min then
                min ← arr[i]
```

$minimum_{index} \leftarrow i$
end if
end for
$X[minimum_{index}] = img[block_{start} : block_{end}]$
$X[minimum_{index}] \leftarrow X[minimum_{index}] \oplus (xor_{pointer})$
$xor_{pointer} \leftarrow xor_{pointer} + minimum_{index}$
$block_{start} \leftarrow block_{start} + block_{size}$
$block_{end} \leftarrow block_{end} + block_{size}$
end while
for $i \leftarrow 0; i \leq matrix.shape[0]; i++$ **do**
$m'.append(X[i])$
end for
$encrypted_{image} \leftarrow m'$

2.3.3. Decryption

$x \leftarrow [\,]$
for $i \leftarrow 0; i \leq matrix.shape[0]; i++$ **do**
$x.append([])$
end for
$block_{size} \leftarrow img.shape[0]/cost_{matrix}.shape[0]$
$block_{start} \leftarrow 0$
$block_{end} \leftarrow block_{size}$
$xor_{pointer} \leftarrow 1$
$visited_{nodes} \leftarrow [\,]$
$i \leftarrow 0$
$pointer \leftarrow 0$
while size(visited_nodes)! = size(x) **do**
$arr \leftarrow matrix[minimum_{index}]$
min $\leftarrow 0$
for $i \leftarrow 0; i \leq size(arr); i++$ **do**
if $i! = minimum_{index}$ and $arr[i] <$ min **then**
min $\leftarrow arr[i]$
$minimum_{index} \leftarrow i$
end if
end for
$block_{end} \leftarrow minimum_{index} \times block_{size}$
$block_{start} \leftarrow block_{end} - block_{size}$
$X[pointer] = img[block_{start} : block_{end}]$
$X[pointer] \leftarrow X[pointer] \oplus (xor_{pointer})$
$xor_{pointer} \leftarrow xor_{pointer} + minimum_{index}$
$pointer \leftarrow pointer + 1$

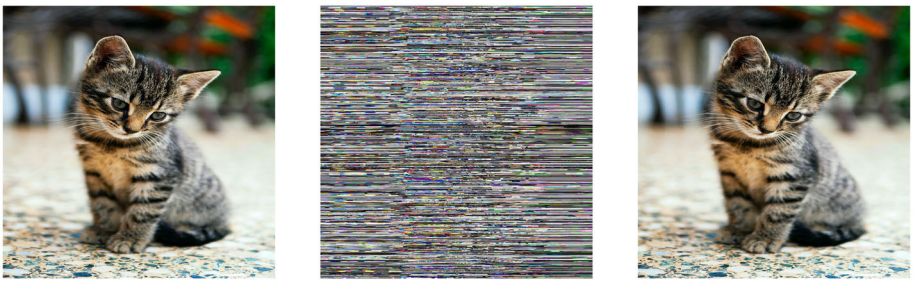
end while
for $i \leftarrow 0; i \leq matrix.shape[0]; i++$ **do**
    $m''.append(X[i])$
end for
$decrypted_{image} \leftarrow m''$

3. Results and discussion

The SWF method was tested on various images to evaluate how well it encrypts various images. First, the author takes images from various domains and uses the SWF encryption technique. The author then uses the SWF decryption algorithm on the encrypted image to test how well it decrypted and if any image was lost. Figures 5(b), 6(b), & 7(b) show that the encrypted images were very noisy to determine the shape, size, or color of the original image. Moreover, these results also indicate that the decryption works as expected and that the original images are retrieved, as shown in Figures 5(c), 6(c), & 7(c).

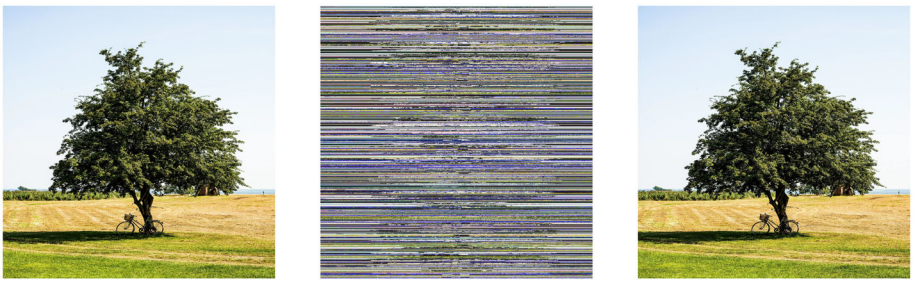
4. Evaluation and analysis

The author assesses the SWF algorithm's performance and security in this section. Only a cursory examination of this new cryptography technique



(a) (b) (c)

Figure 5. (a) Original cat image (b) encrypted cat image (c) decrypted cat image.



(a) (b) (c)

Figure 6. (a) Original tree image (b) encrypted tree image (c) decrypted tree image.

(a) (b) (c)

Figure 7. (a) Original cameraman image (b) encrypted cameraman image (c) decrypted cameraman image.

may be insufficient to assess it effectively. The advantages of this image encryption technique can be assessed using a range of analysis and performance parameters frequently used for image encryption algorithms. The author evaluates the SWF algorithm based on its performance in the key sensitivity, recoverability tests, key space analysis, histogram analysis, and time complexity.

4.1. Time complexity

The Time Complexity of an algorithm is used to determine the amount of time required by the algorithm to perform its operations on the input data provided to it. When comparing two algorithms that provide the same output for the same input, the algorithm with the lower time complexity is considered to be the better.

In this subsection, the author analyses the time complexity of the SWF algorithm.

When the input of an algorithm tends toward a specific value or a restricting value, asymptotic notations are used to quantitatively depict the program's running time. The author uses Big O notation to compute the algorithm's time complexity. Big-O is a type of mathematical notation that expresses an algorithm's worst-case efficiency in terms of input size. In addition, big O notation is used to illustrate algorithm performance and represent how efficiently one algorithm performs compared to another.

4.1.1. Time complexity of algorithm

Let the columns and rows of the matrix produced from the source image be N and M; therefore, a $M \times N$ matrix is obtained. Because the algorithm allocates four rows to each node in the weighted matrix during traversal, this approach has an $O(M/4)$ time complexity. Alternatively, the time complexity with another method can also be calculated using another method.

The weighted matrix can be derived from key1, key2, and key3; let X be the number of rows and Y be the number of columns of the weighted matrix. As the value of X and Y is equal to $M/4$, the time complexity of this algorithm is $O(M/X)$.

After creating the output matrix, XOR values of the image are present in each row of the source image. Therefore, the time complexity of this operation would be $O(M)$.

Therefore the time complexity of the SWF algorithm is $O(M + M/4)$

4.1.2. Time complexity of weighted matrix generation

Assume K numbers are extracted from the image and stored in a data structure. The structure's K values are given to key1 twice, resulting in $2K$ values; this process has a time complexity of $O(2K)$.

Now $3K$ values are present in the structure; these values are passed to key2 twice and generate $3K$ more values; this operation has a time complexity of $O(6K)$ Finally, a structure with 450 or $(6K)$ values is obtained. Assuming that rotating each row takes $O(1)$ time, use key3 to shift each value in the structure. This operation costs $O(9K)$. As a result, the overall cost of creating the weighted matrix is $O(2K + 6K + 9K)$.

Table 1 contains the time taken to encrypt RGB images in seconds, while Table 2 contains the time taken in seconds to decrypt the images ecrypted in Table 1.

Table 1. Processing time analysis for encryption.

Image (1800 × 1800) pixels	Processing time (in seconds)
Cat	3.39
Tree	3.21
Cameraman	3.66
Cycle	3.05
Beach	4.02
Cloud	3.40
Wolf	2.84
Butterfly	3.79
Painting	4.20

Table 2. Processing time analysis for decryption.

Image (1800 × 1800) pixels	Processing time (in seconds)
Cat	3.79
Tree	4.01
Cameraman	4.08
Cycle	3.36
Beach	4.12
Cloud	3.68
Wolf	2.75
Butterfly	4.19
Painting	4.21

4.2. Key space analysis and its security implications

The amount of feasible, valid, and distinct keys that can be utilized in a particular method is specified by the key space value; a larger key space value suggests that this system is secure against attackers.

Three different keys are used in the SWF algorithm; these keys are in the form of polynomial equations, which can have any combination of coefficient, power, and constant, which would be of the encrypter's choice.

To analyze the SWF algorithm, the author assumes Key1 contains two variables, x and y, with power and constant values ranging from 0 to 10. Assume that the equation also has a constant with values ranging from 1 to 10^{17}.

Since there are three wheels namely coefficient, power and constant with possible values ranging between 0 to 9 for power and between 1 to 10^{17} for coefficients and constant, there would be $10 \times 10 \times 10^{17} \times 10^{17} \times 10^{17}$ combinations or a key space size of 10^{53} for key 1.

Similarly, the algorithm has key 2 with a similar key space size of 10^{53}.

For key 3, assume that key 3 has one variable and two constants. The coefficient of variable and value of constant can range between 1 and 10^{17}. Likewise, the power of the variable can range between 1 and 10. This would result in a key space size of $10^{17} \times 10^{17} \times 10$ i.e. 10^{35}.

Therefore the total key space size of all three keys of the SWF algorithm is $10^{53} \times 10^{53} \times 10^{35}$ which would sum up to 10^{141}. After analysis, it can be deduced that key combinations of the SWF algorithm would have a space size of 10^{141}. This ensures that the algorithm is secure in defending itself against brute force attacks. The encrypter can increase variables, power, and constants according to their requirements which would further increase the key space size of the algorithm.

4.2.1. Security analysis

According to (Pramanik et al., 2022), a cryptographic algorithm is said to be secure if the equation is satisfied:

$$P[\mathcal{S} = s|\mathcal{C} = c] = P[\mathcal{S} = s] \tag{2}$$

where S represents all possible source images, and C represents all possible encrypted images. $P\ [S=s]$ estimates the probability of the source image being a particular image s before seeing the encrypted image. Furthermore, $P\ [S=s\ —\ C=c]$ estimates the probability of the source image being a particular image s after knowing that the encrypted image is c.

This section provides a mathematical proof of the security of the algorithm, where $P[]$ stands for probability, $E(S,K)$ denotes the encryption function that takes two inputs, $s \in \mathcal{S}$: Original image, $k \in \mathcal{K}$: encrypting keys. Although there are three equations acting as the keys, this section assumes

that the keys are represented by K. For an arbitrary encrypted image $c \in \mathcal{C}$ and the original image $s \in \mathcal{S}$, we have:

$$P[C = c|S = s] = \frac{P[C = c, S = s]}{Pr[S = s]} \tag{3}$$

Since the encryption function E is bijective i.e.,

$$(S1, K1) \neq E(S2, K2), \forall (S1, K1) \neq (S2, K2), \tag{4}$$

Hence

$$P[C = c, S = s] = Pr[K = k, S = s] \tag{5}$$

The independence of K and S, implies

$$\begin{aligned} &P[C = c, S = s] = P[K = k].P[S = s] \\ &P[C = c|S = s] = \frac{P[K = k].P[S = s]}{P[S = s]} \\ &= P[K = k] = 10^{-141} \end{aligned} \tag{6}$$

Hence,

$$\begin{aligned} &P[C = c] = \textstyle\sum_{s \in S} P[C = c|S = s)].P[S = s] \\ &= 10^{-141} \textstyle\sum_{s \in S} P[S = s] \\ &= 10^{-141} \\ &= P[S = s] \end{aligned} \tag{7}$$

$$\begin{aligned} &P[S = s|C = c] = \frac{P[C = c|S = s].P[S = s]}{P[C = c]} \\ &= \frac{10^{-141}.P[S = s]}{10^{-141}} \\ &= P[S = s] \end{aligned} \tag{8}$$

Therefore, this proves the security of our algorithm.

4.3. Histogram analysis

In image processing, an image's histogram references a histogram of the intensity of the pixel values. This creates a histogram that shows how many pixels are present in an image at each intensity value. Color histograms are adaptable structures that can be created from images in a color space of any dimension. An image's histogram is created by discretizing the image's colors into several bins and calculating the number of image pixels in each bin.

Consider the source image to be a beach image. The SWF algorithm is then used to encrypt the original beach image. Figures 8(d,e) depict the histograms indicating the intensity distribution for the source picture and the encrypted image, respectively.

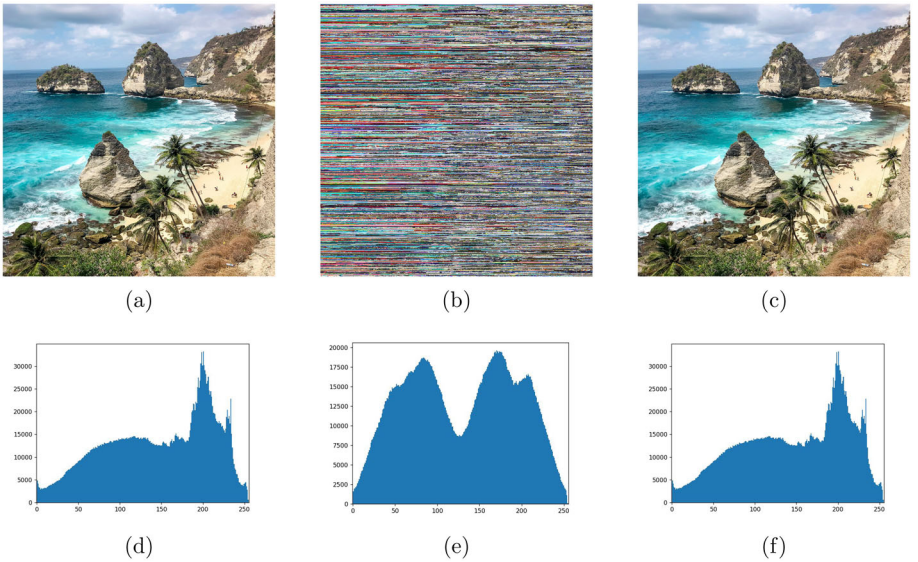


Figure 8. Results of histogram analysis, (a) original Beach image (b) encrypted beach image (c) decrypted beach image (d) histogram of beach image (e) histogram of encrypted beach image (f) histogram of decrypted beach image.

The encrypted image's histogram does not depict a completely uniform intensity distribution. The encrypted image has an intensity distribution between 0 and 17,500, whereas the original beach image has an intensity distribution between 0 and 30,000. It is observed that the encrypted image has a much more uniform distribution than the original image. This indicates that attackers must go through many tedious operations to break the SWF encryption algorithm.

4.4. Key sensitivity test

High sensitivity to key combination changes is required for a secure cryptography algorithm. For example, the author assumes that the SWF algorithm has an integer value key. The algorithm would output two different encrypted images for the same source image even if the key used throughout both encryption processes were modified by a single digit only due to its high sensitivity to changes in key combinations. Figure 9(a) displays the original source image.

For encryption displayed in image 9(b), following key values have been used:

$$\begin{aligned} \text{Key } 1 &= 2x + 3y \\ \text{Key } 2 &= x^2 + y^2 \\ \text{Key } 3 &= 3x^4 - 7x^3 - 12x^2 \end{aligned}$$

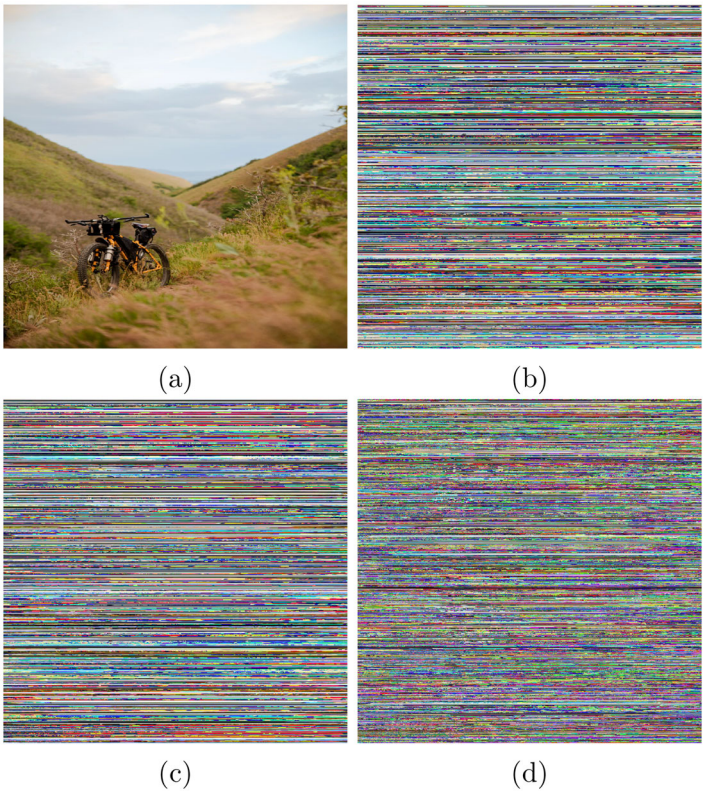


Figure 9. Key sensitivity test for image encryption (a) original cycle image (b) the encrypted cycle image using keys Key $1 = 2x + 3y$, Key $2 = x2 + y2$, Key $3 = 3 \times 4 - 7 \times 3 - 12 \times 2$ (c) the encrypted cycle image using keys Key $1 = 2x + 3y$, Key $2 = x2 + y2$, Key $3 = 4 \times 4 - 7 \times 3 - 12 \times 2$(d) the difference between Figures 9(b) and 9(c).

Now change just one integer value in Key 3 and set them to the following values

$$\begin{aligned} \text{Key } 1 &= 2x + 3y \\ \text{Key } 2 &= x^2 + y^2 \\ \text{Key } 3 &= 4x^4 - 7x^3 - 12x^2 \end{aligned}$$

It can be seen that the slightest changes in Key 3 significantly impact the encrypted image in Figure 9(c). Figure 9(d) depicts the difference between the two encrypted images and indicates a significant variation between the two images.

Based on the findings of this test, it can be said that even little modifications to the key values of the SWF algorithm significantly affect the outcomes of the encryption process. It should be emphasized that all other this algorithm's adjustable parameters were kept constant during both encryption operations. As a result, it can be inferred that the encryption

process of the suggested method is very key sensitive, which increases the advantages of applying this algorithm in real-world applications.

4.5. Recoverability test

Recovering the entirety of the source image after applying a decryption algorithm is an essential aspect that should be present in any cryptography algorithm. After the decryption technique has been used to decrypt the encrypted image, the recoverability test is used to evaluate if all of the source images have been recovered. It computes the contrast between the decrypted image and the source image. If the resultant image of the mentioned computation is black, it will prove that the entire source image has been successfully recovered from the encrypted image. This would imply that no data was lost, and the algorithm would pass the recoverability test.

Figure 10(a) displays Wolf, the original source image, and Figure 10(b) displays the decrypted image produced by the SWF technique. The distinction between Figure 10(a,b) is depicted in Figure 10(c). Figure 10(c) is an entirely black image which implies that 100% of the source image is successfully recovered after decryption is performed using the SWF algorithm. Figure 11 shows the same results for a different image. These results prove that the SWF algorithm successfully passes the recoverability test.

4.6. Noise resistance analysis

During transmission, an image may suffer some noise infection. An algorithm must decrypt loss-induced images with the least amount of distortion to handle such data loss, whether it occurs during data transfer or is intentionally introduced noise. To test this property of the SWF algorithm, the noise was deliberately added to two encrypted images, and their corresponding decrypted images were generated. Figure 12(a,e) represent the original butterfly and painting image, respectively. The encrypted images,

(a) (b) (c)

Figure 10. (a) Original wolf image (b) decrypted wolf image (c) image depicting difference in Figures 10(a) and 10(b).

Figure 11. (a) Original cloud image (b) decrypted cloud image (c) image depicting difference in Figures 11(a) and 11(b).

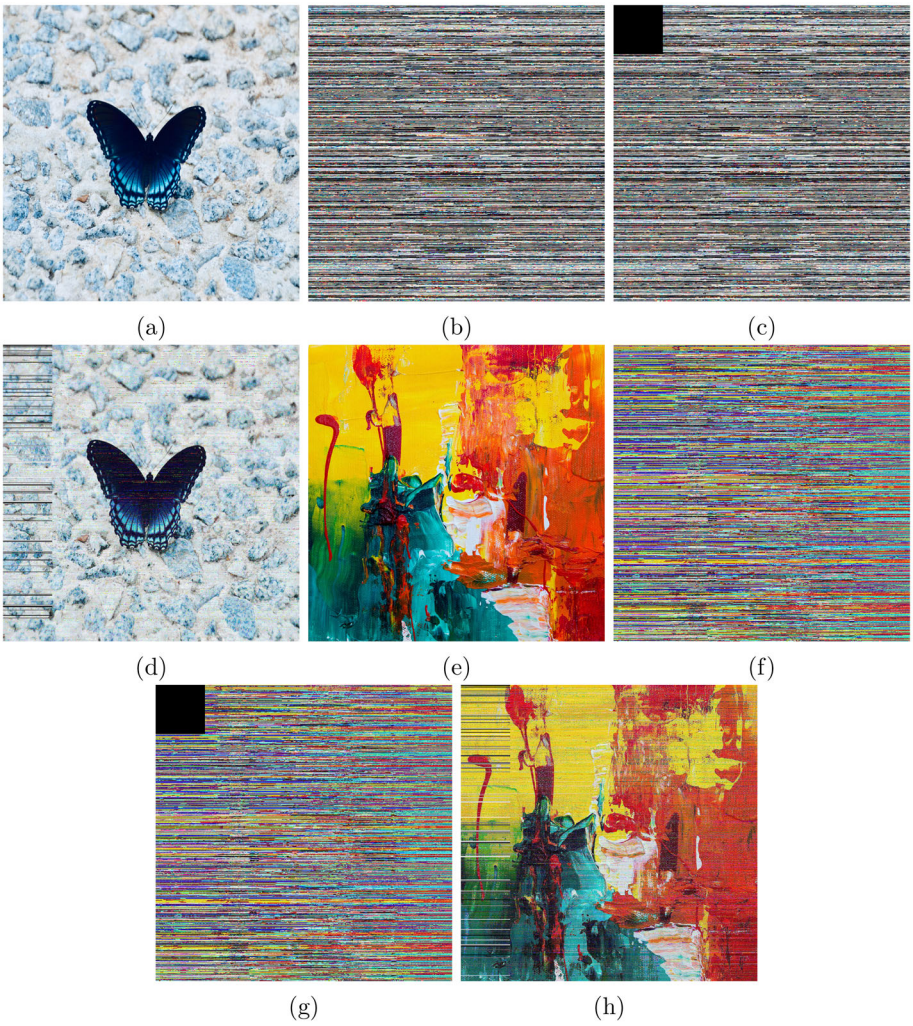


Figure 12. Result of Noise-resistance analysis, (a) original butterfly image (b) encrypted butterfly image (c) encrypted butterfly image with noise (d) decrypted butterfly image (e) original painting image (f) encrypted painting image (g) encrypted painting image with noise (h) decrypted painting image.

as shown in Figure 12(c,g), were given a noise addition of $300 \times 300 = 90{,}000$ pixels; the resulting decoded images are displayed in Figure 12(d,h), respectively. The results indicate that noise would not enormously affect the SWF algorithm, and a somewhat homogeneous image is obtained even after the encrypted image might get affected by noise.

4.7. Comparison with other algorithms

In this section, the author compares the SWF algorithm with two famous algorithms, Hill Cipher and RSA Algorithm.

4.7.1. Hill Cipher

The Hill Cipher is a symmetric cipher invented by Lester S. Hill. It uses the concept of matrix multiplication. The equations used for encryption and decryption are given below.

Encryption:

$$E(K, P) = (K * P) mod 256 \tag{9}$$

where K is the key and P is the matrix to be eliminated.

Decryption:

$$D(K, C) = (K^{-1} * C) mod 256 \tag{10}$$

where K^{-1} is the inverse of the key matrix and C is the encrypted matrix.

4.7.2. RSA algorithm

The RSA is an asymmetrical encryption algorithm invented by Rivest, Shamir, and Adelman (RSA). It uses two keys, a public key and a private key, where the public key is used to encrypt, and the private key is used to decrypt. The equations used for encryption and decryption are given below.

Encryption:

$$C = m^e mod n \tag{11}$$

where (e,n) is the public key.

Decryption:

$$M = c^d mod n \tag{12}$$

where (d,n) is the private key.

The SWF algorithm, Hill Cipher and RSA algorithm are all used to encrypt the same grayscale image. It is found that the image was encrypted using the Hill Cipher in 1.56, RSA in 2.07, and the SWF technique in 1.34 s. Additionally, the Hill Cipher took 1.78 s, RSA took 2.95 s, and the

Table 3. Comparison of SWF algorithm with Hill Cipher and RSA algorithm.

Parameter	Hill Cipher	RSA	SWF Algorithm
Time taken(in seconds) for encrypting painting image (Gray Scale)	1.56	2.7	1.34
Time taken(in seconds) for decrypting painting image (Gray Scale)	1.78	2.95	1.47

SWF algorithm took 1.47 s to decrypt the encrypted image. These findings are presented in tabular form in Table 3. After comparing the SWF method to the Hill Cipher and the RSA algorithm, it can be concluded that the SWF algorithm takes less time to encrypt and decrypt the same image.

4.8. Advantages

The SWF method is presented as a novel method for effective image cryptography. We come to the conclusion that using the suggested strategy has a lot of advantages, as evidenced by the outcomes and results of applying the SWF algorithm to different image kinds and conducting analysis and assessment of the method. This algorithm uses the concept of shortest weight first on a weighted graph which is easy to understand and implement. The SWF algorithm's key space is sufficiently large to fend off brute-force attacks. Changes in security key values have a significant impact on the algorithm's encryption process. After performing histogram analysis, it is seen that there is no discernible difference between the source image and the encrypted image. This indicates that the SWF technique fully recovers the original picture from the decrypted image. The SWF algorithm can be applied to images from various industries, including telemedicine, multimedia systems, military communication, and internet communication. The SWF algorithm provides flexibility while choosing polynomials for weighted matrix generation. The SWF algorithm's time complexity is significantly less compared to many currently used image cryptography techniques. The SWF algorithm does not rely on external keys during its encryption and decryption process.

5. Conclusion

This research presented a new image encryption algorithm that is simple to comprehend and implement and is based on graph theory's shortest weight first approach. This algorithm has various applications; it can transmit images from various domains like humans, animals, nature, medicine, art, and astronomy, as seen in the examples given throughout this paper. It is of utmost importance that the encryption-decryption process does not introduce any loss in the image as it could have severe consequences. Since this algorithm does not introduce any loss after decryption, it can be used to encrypt images during transmission. When this algorithm is analyzed in

detail, as done in section 4, it is seen that the key space of this algorithm is vast, and therefore it has good protection against brute force attacks. Furthermore, the recoverability test proves the algorithm's ability to decrypt the images without any loss. The noise resistance test shows that noise infection will not affect its ability to decrypt images with the least amount of distortion possible. All these observations show that this algorithm can be used in the real world and has many practical applications.

Various images have been used to test this algorithm; it has successfully decrypted them without causing any loss. It also allows the user complete control over a key selection from an ample key space. As a result, this technique is effective in encrypting and decrypting images.

This method can be used to encrypt not only photos but also the visual content of videos. This is because videos consist of frames, so the entire video can be protected by employing this approach to encrypt each frame. This algorithm organized the graph's nodes using the shortest weight first method; however, one might apply a different method, analyze it, and see if the results are superior to SWF's.

ORCID

Ramchandra Mangrulkar http://orcid.org/0000-0002-9020-0713

References

Abduljaleel, I. Q., Abdul-Ghani, S. A., & Naji, H. Z. (2021). An image of encryption algorithm using graph theory and speech signal key generation. *Journal of Physics: Conference Series*, *1804*(1), 012005. https://doi.org/10.1088/1742-6596/1804/1/012005

Ahmad, M., & Shamsher Alam, M. (2009). A new algorithm of encryption and decryption of images using chaotic mapping. *International Journal on Computer Science and Engineering*, *2*(1), 46–50. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.208.6906&rep=rep1&type=pdf

Ak, M., Hanoymak, T., & Selçuk, A. A. (2014). IND-CCA secure encryption based on a Zheng–Seberry scheme. *Journal of Computational and Applied Mathematics*, *259*(2014), 529–535. https://doi.org/10.1016/j.cam.2013.06.042

Anandakumar, S. (2015). Image cryptography using RSA algorithm in network security. *International Journal of Computer Science & Engineering Technology*, *5*(9), 326–330. https://www.ijcset.net/vol5issue9.php

Chen, G., Mao, Y., & Chui, C. K. (2004). A symmetric image encryption scheme based on 3D chaotic cat maps. *Chaos, Solitons & Fractals*, *21*(3), 749–761. https://doi.org/10.1016/j.chaos.2003.12.022

Dey, D., & Paul, S. (2019). Color image encryption using single layer artificial neural network and buffer shuffling. *International Journal of Computer Sciences and Engineering*, *7*(3), 202–211. https://doi.org/10.26438/ijcse/v7i3.202211

Gao, D., Chen, H., & Chang, C.-C. (2020). Plaintext aware encryption in the standard model under the linear Diffie-Hellman knowledge assumption. *International Journal of*

Computational Science and Engineering, *22*(2/3), 270–279. https://doi.org/10.1504/IJCSE.2020.107349

Gopalan, N. P., & Kumaresan, G. (2020). An effective reversible data-hiding in encrypted images using memory cellular automata. *Journal of Applied Security Research*, *15*(1), 96–115. https://doi.org/10.1080/19361610.2019.1672456

Hsieh, M.-Y., Lin, H.-Y., Lai, C.-F., & Li, K.-C. (2011). Secure protocols for data propagation and group communication in vehicular networks. *EURASIP Journal on Wireless Communications and Networking*, *2011*(1), 1. https://doi.org/10.1186/1687-1499-2011-167

Hu, W.-T., Li, M.-C., Guo, C., & Yuan, L.-F. (2015). A reversible steganography scheme of secret image sharing based on cellular automata and least significant bits construction. *Mathematical Problems in Engineering*, *2015*, 849768. https://www.researchgate.net/publication/276090887_A_Reversible_Steganography_Scheme_of_Secret_Image_Sharing_Based_on_Cellular_Automata_and_Least_Significant_Bits_Construction

Jeppiaar, M. R. (2015). A prediction based reversible image steganographic algorithm for JPEG images. *Journal of Applied Security Research*, *10*(3), 362–374. https://doi.org/10.1080/19361610.2015.1038766

Jeyaprakash, H., Kartheeban, K., Sahu, A. K., & Chokkalingam, B. (2022). Data hiding using PVD and improving security using RSA. *Journal of Applied Security Research*, *17*(3), 413–420. https://doi.org/10.1080/19361610.2021.1900692

Joseph, L., Renjit, J. A., & Kumar, P. M. (2013). Dynamic programming based encrypted reversible data hiding in images. *Journal of Applied Security Research*, *8*(4), 467–476. https://doi.org/10.1080/19361610.2013.825753

Kamal, S. T., Hosny, K. M., Elgindy, T. M., Darwish, M. M., & Fouda, M. M. (2021). A new image encryption algorithm for grey and color medical images. *IEEE Access*. *9*(2021), 37855–37865. https://doi.org/10.1109/ACCESS.2021.3063237

Man, Z., Li, J., Di, X., Sheng, Y., & Liu, Z. (2021). Double image encryption algorithm based on neural network and chaos. *Chaos, Solitons & Fractals*, *152*(2021), 111318. https://doi.org/10.1016/j.chaos.2021.111318

Mehta, D., Jha, M., Suhagiya, H., & Mangrulkar, R. (2022). *DieRoll: A unique key generation and encryption technique* (pp. 1–28). *Journal of Applied Security Research*. https://doi.org/10.1080/19361610.2022.2124589

Mhatre, M., Kashid, H., Jain, T., & Chavan, P. (2022). *BCPIS: Blockchain-based Counterfeit Product Identification System* (pp. 1–21). *Journal of Applied Security Research*. https://doi.org/10.1080/19361610.2022.2086784

Mittenthal, L. (2007). Sequencings and directed graphs with applications to cryptography. In *Sequences, subsequences, and consequences* (pp. 70–81). Springer. https://doi.org/10.1007/978-3-540-77404-4_7

Najjar, A., & Saleh, A. L. (2017). *Implementation color-images cryptography using RSA algorithm*. https://www.researchgate.net/publication/321906037_Implementation_Color-Images_Cryptography_Using_RSA_Algorithm

Nandy, N., Banerjee, D., & Pradhan, C. (2021). Color image encryption using DNA based cryptography. *International Journal of Information Technology*, *13*(2), 533–540. https://doi.org/10.1007/s41870-018-0100-9

Omotosho, A., & Emuoyibofarhe, J. (2015). Private key management scheme using image features. *Journal of Applied Security Research*, *10*(4), 543–557. https://doi.org/10.1080/19361610.2015.1069642

Patanwadia, R., & Mangrulkar, R. (2021). Divide and scramble - a recursive image scrambling algorithm utilizing Rubik's cube. In *2021 International Conference on Recent*

Trends on Electronics, Information, Communication & Technology (RTEICT), IEEE (pp. 859–863). https://doi.org/10.1109/RTEICT52294.2021.9573949

Pourasad, Y., Ranjbarzadeh, R., & Mardani, A. (2021). A new algorithm for digital image encryption based on chaos theory. *Entropy*, *23*(3), 341. https://doi.org/10.3390/e23030341

Pramanik, S., Ghonge, M. M., Mangrulkar, R., & Le, D. N. (Eds.). (2022). *Cyber security and digital forensics: Challenges and future trends*. John Wiley & Sons. https://www.wiley.com/en-us/Cyber+Security+and+Digital+Forensics/%3A+Challenges+and+Future+Trends-p-9781119795643

Sakthidasan, K., & Santhosh Krishna, B. V. (2011). A new chaotic algorithm for image encryption and decryption of digital color images. *International Journal of Information and Education Technology* 1(2), 137. http://www.ijiet.org/index.php?m=content&c=index&a=show&catid=23&id=198

Siahaan, A. P. U. (2017). *RC4 technique in visual cryptography RGB image encryption*. https://www.researchgate.net/publication/305268410_RC4_Technique_in_Visual_Cryptography_RGB_Image_Encryption

Storms, J. E. (2016). *An evaluation of the history, demand, and current methods for digital steganography* [PhD dissertation]. Utica College. https://www.proquest.com/openview/8a1fc9d235c06c435f16df2a1bf25274/1?pq-origsite=gscholar&cbl=18750&diss=y

Tahat, N., & Abdallah, E. (2018). Hybrid publicly verifiable authenticated encryption scheme based on chaotic maps and factoring problems. *Journal of Applied Security Research*, *13*(3), 304–314. https://doi.org/10.1080/19361610.2018.1463135