

Research Article

Mohit Dua*, Drishti Makhija, Pilla Yamini Lakshmi Manasa, and Prashant Mishra

3D chaotic map-cosine transformation based approach to video encryption and decryption

<https://doi.org/10.1515/comp-2020-0225>

received August 19, 2020; accepted February 2, 2021

Abstract: Data security is vital for multimedia communication. A number of cryptographic algorithms have been developed for the secure transmission of text and image data. Very few contributions have been made in the area of video encryption because of the large input data size and time constraints. However, due to the massive increase in digital media transfer within networks, the security of video data has become one of the most important features of network reliability. Block encryption techniques and 1D-chaotic maps have been previously used for the process of video encryption. Although the results obtained by using 1D-chaotic maps were quite satisfactory, the approach had many limitations as these maps have less dynamic behavior. To overcome these drawbacks, this article proposes an Intertwining Logistic Map (ILM)-Cosine transformation-based video encryption technique. The first step involved segmenting the input video into multiple frames based on the frames per second (FPS) value and the length of the video. Next, each frame was selected, and the correlation among the pixels was reduced by a process called permutation/scrambling. In addition, each frame was rotated by 90° in the anticlockwise direction to induce more randomness into the encryption process. Furthermore, by using an approach called the random order substitution technique, changes were made in each of the images, row-wise and column-wise. Finally, all the encrypted frames were

jumbled according to a frame selection key and were joined to generate an encrypted video, which was the output delivered to the user. The efficiency of this method was tested based on the state of various parameters like Entropy, Unified Average Change in Intensity (UACI), and correlation coefficient (CC). The presented approach also decrypts the encrypted video, and the decryption quality was checked using parameters such as mean square error (MSE) and peak signal-to-noise ratio (PSNR).

Keywords: ILM, chaotic maps, UACI, Entropy, PSNR

1 Introduction

Rapid innovation in the wireless communication field has created many challenges to transfer data in a secure manner. Due to these advancements, there has been a significant increase in the transmission of images and videos through the public network. Also, the introduction of social media has led to the mass sharing of videos on the Internet frequently. Because of this intensification of sharing videos, there is a high demand for data security. Unencrypted data are more vulnerable to attacks. Therefore, there is a need for the data to be transmitted securely through any network, and the method called encryption ensures this [1–3]. There are various encryption algorithms used to encrypt/decrypt data. Most of these algorithms are focused on image and text data. Because of the large input size and huge time constraints, a less significant development was made when it comes to video encryption [4–13]. The algorithms that produced some good results in the past are explained in the coming paragraphs; however, in this era of social media, better encryption techniques are required.

In 1998, Shi and Bhargava [6] proposed an efficient MPEG video encryption algorithm that uses a secret key. This secret key is used for changing the sign bits of encoded differential values of motion vectors of B and P pictures randomly. This approach basically conducted three major steps. It encrypts the motion vectors of B and P frames and then encrypts the DC coefficient of

* **Corresponding author: Mohit Dua**, Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana 136119, India, e-mail: er.mohitdua@nitkkr.ac.in, tel: +91-9466588448

Drishti Makhija: Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana 136119, India, e-mail: drishti1374@gmail.com

Pilla Yamini Lakshmi Manasa: Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana 136119, India, e-mail: pylmanu.145@gmail.com

Prashant Mishra: Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana 136119, India, e-mail: pm.vodd@gmail.com

I frame and then encrypts all coefficients of motion vectors and I frame. Experimental results show that this adds overhead to MPEG codec and can be used only for securing video emails applications.

Chiaraluce et al. in ref. [14] proposed a new chaotic algorithm for video encryption. The concept of this algorithm is to use three different chaotic functions. It uses two keys that act as an input for two chaotic functions and the XOR of the output act as input for the third function. The output from this chaotic function is XORed with video data that results in the encrypted video. The security could be increased by using a larger key length.

Li et al. in ref. [15] proposed a new video encryption algorithm for H.264, which is a video encoding standard, which guarantees the security. In this method, the user key is responsible for four major parts: interprediction and intraprediction mode scrambling, encryption of motion vectors, and encryption of transform. The old prediction mode is XORed with a three-digit random sequence to get the new prediction mode. Because of the two-dimensional chaos system, this algorithm works better in terms of security but suffers from the problem of having less effect on entropy.

Raju et al. in ref. [16] proposed a fast and real-time computationally efficient video encryption algorithm. It compresses the video data using Discrete Cosine Transform (DCT). Each video is divided into frames, and DCT is applied on a block of 8×8 size. It uses the RC5 (Rivest Cipher) algorithm for encrypting the DCT coefficients. This approach produces high security and entropy but adds overhead to codec.

Dumbere and Janwe in ref. [17] proposed Advanced Encryption Standard (AES) algorithm for video encryption. It uses MATLAB for the implementation. It encrypts the 128-bit block data in “N” round of encryption based on the key. It produces better results and takes less encryption and decryption time than Data Encryption Standard (DES).

The works proposed in refs. [18–23] are the main motivation behind the proposed approach of this article. The techniques proposed in refs. [18–20] use 1D chaos map, i.e., logistic map. Although a logistic map is simplest to implement and exhibits good chaotic behavior, it suffers from the issues of a blank and stable window. The proposed work of this article is an extension of these works. Ye and Huang in ref. [21] proposed an approach for image encryption using ILM. It is found that results obtained from ILM performs better than LM. In 2018, Hua et al. in ref. [22] proposed a method that coupled the logistic map with sine function that exhibits more complexity and large chaotic range. Recently, in 2019, a work

is proposed in ref. [23], in which a logistic map combined with sine function acts as an input for cosine function to increase the nonlinearity in the chaotic sequence.

This article proposes a fast and secure method of video encryption using 3D ILM with cosine transformation to generate a complex chaotic sequence. The proposed algorithm performs better in terms of security and efficiency and is highly resistant to cyber-attacks due to its nonlinearity. The novelty of this approach is using the 3D ILM cosine transformation for generating the chaotic sequence, which is provided as an input to the cosine function to produce a new chaotic pattern, which is more nonlinear. This article also checks the quality of decrypted video using MSE and PSNR parameters.

The rest of this article is divided as follows: Section 2 describes the preliminaries, i.e., the fundamentals of chaos, LM, and ILM. Section 3 elaborates the proposed architecture. Implementation and experimental setup details are presented in Section 4. Section 5 compares the results with other existing approaches of video encryption. Section 6 discusses the conclusion and future works related to this technique.

2 Preliminaries

This section annotates the fundamentals of chaos theory, logistic maps (LM), intertwining logistic map (ILM), and ILM-Cosine, which were used to generate keys for the proposed video encryption algorithm.

2.1 Chaos theory

Chaos theory is a mathematical theory [24,25] that is still a part of the research. It belongs to the dynamics, a field of physics that concerns the motion of objects when any force is applied. It is also the study of how simple patterns can be generated using the complicated behavior of numbers. The basic structure of chaos is divided into two parts, i.e., permutation and diffusion. Permutation is defined as the scrambling of the quantities in which the values are placed randomly. On the other hand, diffusion is a kind of substitution that is used for removing the redundancy from the chaos.

Later, in the 1900, Lorenz used the term chaos for the first time [26]. He studied chaos theory in contextual weather systems. His studies prove that chaos theory can make decisions in complex environments. Because

of handling the unpredictability in complex systems, chaos is used for encrypting sensitive information. In 1997, Fridrich [27] proposed an image encryption algorithm based on chaotic maps. Since then chaos has been used for encrypting digital data and its secure transmission. The unpredictability and intractability of chaos make it useful for this proposed algorithm for video encryption.

2.2 Logistic map

Logistic map (LM) is a two-degree polynomial mapping chaos function. It is the most popular and simplest of all the chaotic functions. It is used to generate chaotic behavior [28] from nonlinear dynamic equations. The mathematical representation of this function is defined as follows:

$$U_{s+1} = \eta * U_s(1 - U_s), \quad (1)$$

where the range of U_s varies in $(0, 1]$ that represents the value at sth position in the sequence and is a control parameter that is responsible for complete the chaotic sequence in the range $[3.57, 4]$.

Although LM has good ergodic nature that limits it to depend on initial conditions, it is sensitive to one control parameter. Due to this reason, one-dimensional logistic map extended to two dimensional [29], which is represented by the following equations:

$$U_{s+1} = \eta_1 * U_s(1 - U_s) + \varepsilon_1 * (V_s)^2, \quad (2)$$

$$V_{s+1} = \eta_2 * V_s(1 - V_s) + \varepsilon_2((U_s)^2 + U_s V_s). \quad (3)$$

These equations are responsible for generating chaotic sequences in the range $(0, 1]$ represented by U and V . Here, U_s and V_s represent the value at sth position in two different dimensions. This chaos map remains in a chaotic state when $2.75 < \eta_1 \leq 3.4$, $2.75 < \eta_2 \leq 3.45$, $0.15 < \varepsilon_1 \leq 0.21$, and $0.13 < \varepsilon_2 \leq 0.15$. With further technical developments, this is extended to three-dimensional chaotic sequences [30,31]. The equations for 3D chaotic maps are described as follows:

$$U_{s+1} = \eta * U_s(1 - U_s) + (V_s)^2 U_s + (W_s)^3, \quad (4)$$

$$V_{s+1} = \eta * V_s(1 - V_s) + (W_s)^2 V_s + (U_s)^3, \quad (5)$$

$$W_{s+1} = \eta * W_s(1 - W_s) + (U_s)^2 W_s + (V_s)^3. \quad (6)$$

Here, U_s , V_s , and W_s represent the value at sth position in three dimensions. These equations exhibit nonlinear system when U_0 , V_0 , and W_0 are in $[0, 1]$ and $0.53 < U_0 < 3.81$, $0 < V_0 < 0.022$, and $0 < W_0 < 0.015$. These generated sequences have good cross-relation and

auto-correlation, but they also have some drawbacks. LM suffers from the problem of stable windows, blank windows [32], and nonuniform sequence distribution. Figure 1 shows the problem of blank windows in LMs. To overcome these boundaries of LMs, ILMs came into the picture.

2.3 ILM

In 2014, Wang and Xu [33] proposed an intertwining relation between different LM sequences, which are nonlinear. Since the Lyapunov exponent of ILM is always positive in comparison to LM, it indicates that ILM has more dynamic behavior than LM [34]. The equations for ILM sequence are expressed as follows:

$$U_{s+1} = [\eta * \sigma * V_s * (1 - U_s) + W_s] \text{ Mod } 1, \quad (7)$$

$$V_{s+1} = [\eta * \vartheta * V_s + W_s * (1 + U_s + 12)] \text{ Mod } 1, \quad (8)$$

$$W_{s+1} = [\eta * (V_s + 1 + U_s + 1 + \kappa) * \sin(W_s)] \text{ Mod } 1, \quad (9)$$

where η is in range of $[0, 4)$, $\sigma > 33.5$, $\vartheta > 37.9$, and $\kappa > 35.7$ for exhibiting the chaotic behavior.

As shown in Figure 2, chaotic sequences generated by ILM are uniformly distributed. This uniform distribution results in the removal of LM's demerits, i.e., uneven key distribution, blank window, and stable window.

Figure 3 shows the Lyapunov exponent for LM and ILM methods. It can be observed that the Lyapunov exponent of ILM is never negative and is more uniformly distributed.

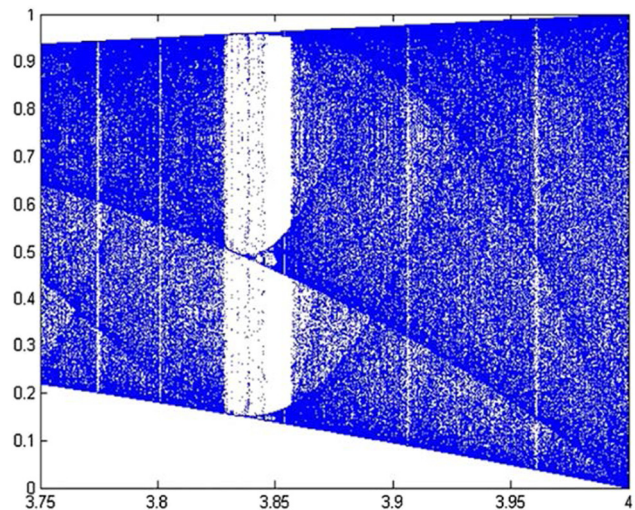


Figure 1: Sequence distribution in LM showing blank window with η on X-axis and U on Y-axis [33].

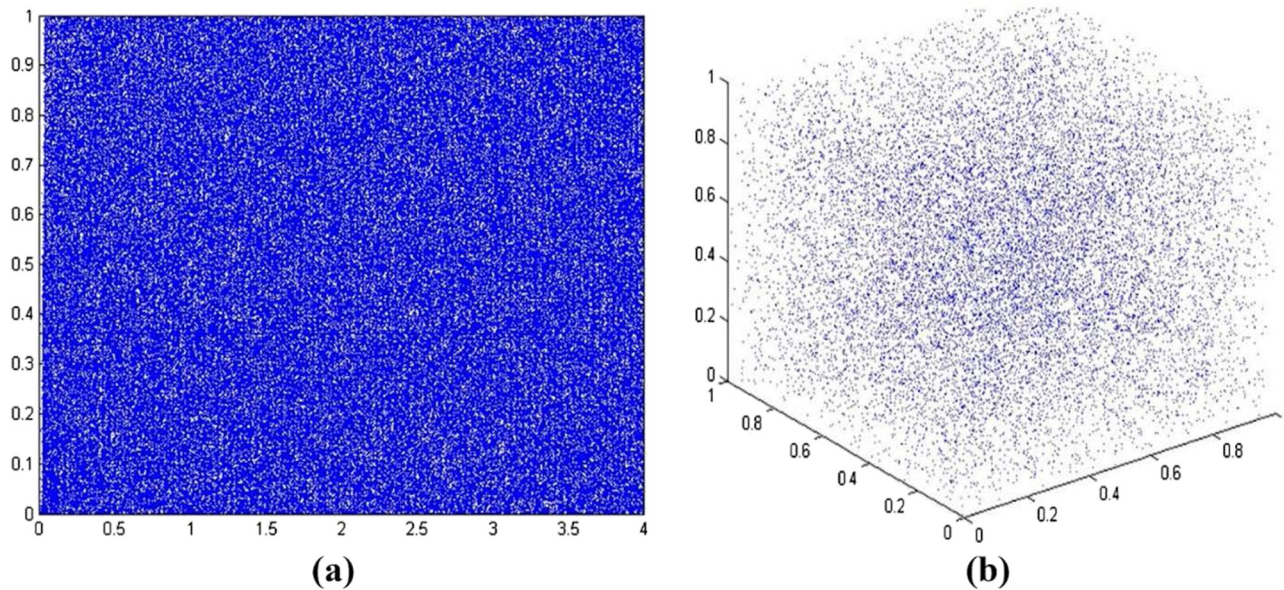


Figure 2: ILM: (a) single-sequence distribution (U , V , or W) and (b) three-sequences distribution (U , V , and W) [33].

2.4 ILM with cosine function (ILM-cosine)

ILM indicates the dynamic behavior in the output. But to increase the efficiency of encryption, ILM is combined with cosine function in the proposed approach. The cosine function is used to increase the content of non-linearity in the output produced by ILM [35]. The equations for ILM-cosine are expressed as follows:

$$U_{s+1} = \cos([\eta * \sigma * V_s * (1 - U_s) + W_s] \text{ Mod } 1 + \vartheta), \quad (10)$$

$$V_{s+1} = \cos([\eta * \vartheta * V_s + W_s * (1 + U_s + 12)] \text{ Mod } 1 + \vartheta), \quad (11)$$

$$W_{s+1} = \cos([\eta * (V_s + 1 + U_s + 1 + \kappa) * \sin(W_s)] \text{ Mod } 1 + \vartheta). \quad (12)$$

3 Proposed architecture

This section proposes a new symmetric key encryption and decryption method for video data. Chaos-based cryptographic algorithms are used to make it difficult for unauthorized users to break the encryption. The encryption process starts by dividing the input video into multiple frames. Each of these frames go through a series of steps, i.e., permutation, rotation, diffusion to produce

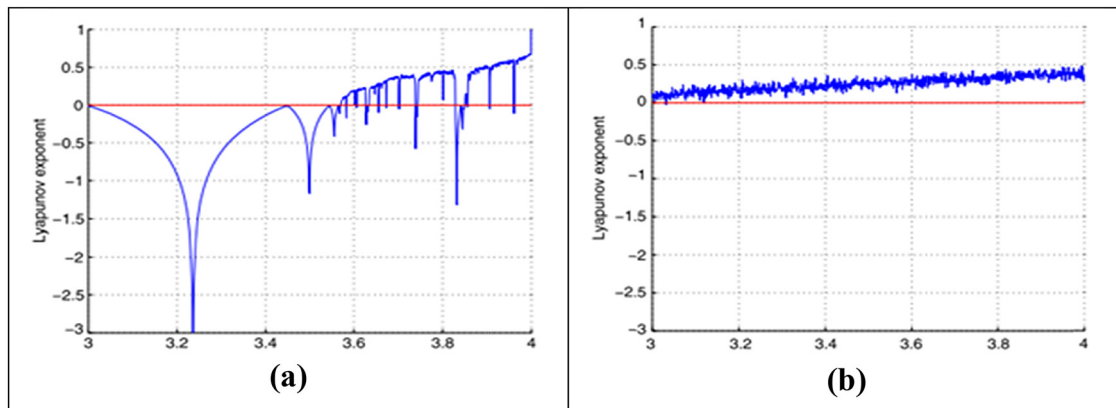


Figure 3: Lyapunov exponents of (a) LM and (b) ILM [33].

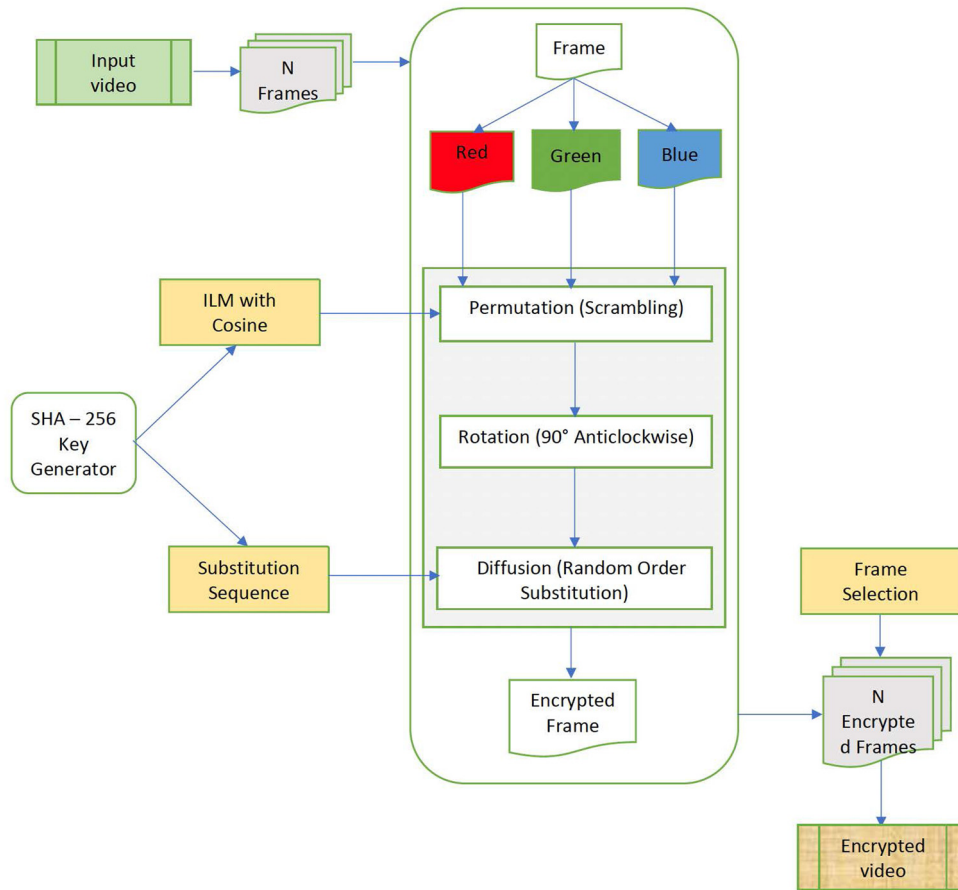


Figure 4: Encryption process of the proposed approach.

encrypted frames. The encrypted frames are jumbled using a frame selection key before joining them to form the final encrypted video. This encrypted video can be transmitted securely as any unauthorized user cannot access it without the keys. The decryption process involves splitting the encrypted video, rearranging the frames according to the FS key, and performing antisubstitution, rotation, and descrambling on each of them to generate decrypted frames. These decrypted frames are merged together to generate the decrypted video.

3.1 Encryption

As soon as a video (in mp4 format) is chosen for encryption, it is split into a number of frames. The duration and FPS value of the video determine the number of frames. The steps followed for encrypting the video are shown in Figure 4. The encryption process is performed on each of the frames iteratively until all the frames are encrypted. Each frame is split into three 2D matrices that represent

its red, green, and blue components. An example of this process is shown in Figure 5.

3.1.1 Key generation

The secret keys required for permutation are generated using a keyless hash function called Secure Hash Algorithm-256 (SHA-256) [36,37]. It converts messages of any size into a hash of fixed length (256 bits) and provides an additional advantage that getting a collision is computationally impossible. It is a new cryptographically secure one-way hash function that falls under the category of SHA 2. SHA 2 is an improved version of SHA 0 and SHA 1.

SHA-256 function is used to generate the seeds of length equal to the key length. Whatever be the length of input information, this algorithm breaks the data into 64 bytes or 512 bits and generates a 256-bit hash seed after cryptographic mixing. Three such seeds are generated for the R, G, and B components of each frame. It is resistant to all kinds of attacks identified till date and thus is used to secure highly sensitive data.

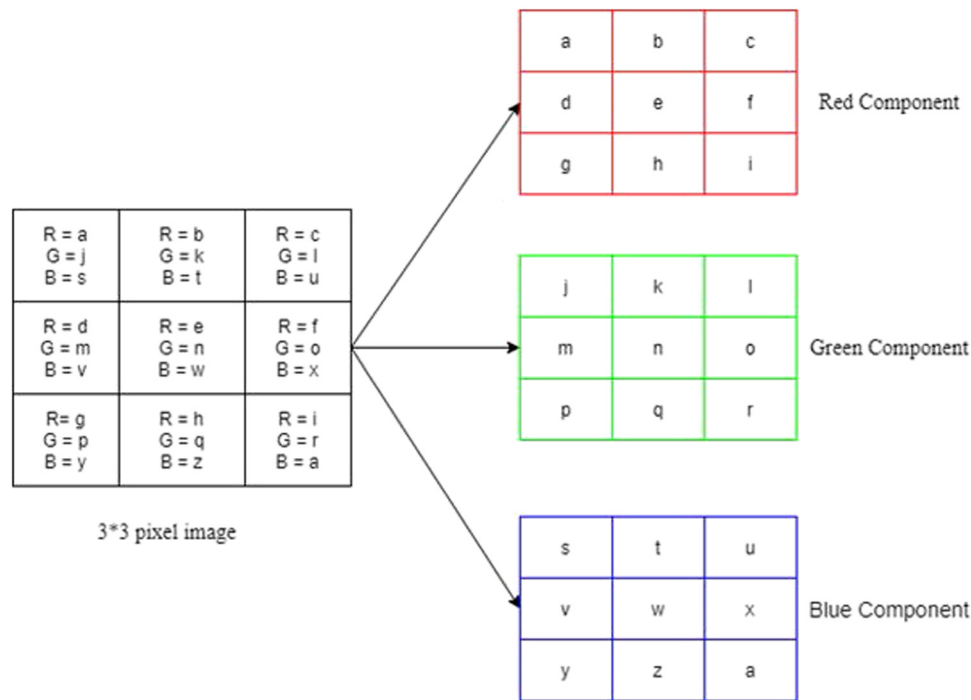


Figure 5: Splitting an image into its R, G, and B components.

3.1.2 Permutation

The process of jumbling the position of pixels in a frame is called permutation or scrambling. This interchanging of pixels helps to reduce the correlation among them. Let A and B be the dimensions of the input image, and I be the size of block, which is calculated as follows:

$$I = \min\{\lfloor \sqrt{A} \rfloor, \lfloor \sqrt{B} \rfloor\}. \quad (13)$$

Then, permutation can be done on the matrix of size $I^2 \times I^2$. Each frame is divided into I^2 blocks. By using the chaotic sequence generated by the key generator, the pixels of each row of the block are shuffled into other blocks. In the next step, another chaotic sequence is used to jumble the position of pixels column-wise. This scrambling of pixels reduces the association between them which is considered to be a feature of a good encryption technique. The following steps are followed for permutation:

1. I is calculated using equation (13) and an image of size $I^2 \times I^2$ pixels is extracted from the frame.
2. Four chaotic sequences, say P, Q, R, and S of length I^2 are generated by the key generator and are sorted.
3. After sorting these sequences, four index vectors are generated, one from each of them. Let them be labeled as In P, In Q, In R, and In S.

4. Two blank matrices L and M are generated with dimensions $I^2 \times I^2$. The matrices L and M are initially filled with the values present in the columns of In P and In R, respectively, and the elements present in L and M are shifted according to the elements of In Q and In S, respectively.
5. The value of row(r) is initialized to 1. Let the value present in the r th row and c th column of L matrix (i.e., (L_r, c) th value) be "x." The (r, c) th pixel of the frame under consideration is replaced by the value in x th row and c th column of M matrix (i.e., M_x, c).
6. Step 5 is done iteratively for all values of "r" in the range $[2, I^2]$.

3.1.3 Rotation

Even after permutation, the position of many pixels in the matrix remains unchanged. This is because only those pixels that are in the range $I^2 \times I^2$ are permuted. To induce more randomness, the entire matrix is rotated by 90° in the anticlockwise direction. This rotates not only the entire frame but also every pixel in it. It is preferable to rotate the matrix in multiples of 90° , as rotating by any other intermediate angles results in a tilted frame, which is unfit for computation.

3.1.4 Diffusion

Diffusion or substitution is the process of spreading variation throughout the matrix by replacing the existing values with new ones. The changes are made in the matrix row-wise and column-wise according to general rules of diffusion. Following these preestablished rules makes the system more susceptible to attacks. To overcome this drawback, random order substitution technique is used, and the equations used to implement the same are as follows:

$$E_{In_r, c, c} = \begin{cases} (L_{In_r, c, c} + L_{In, N} + \lfloor 2^{32} * P_{In_r, c, c} \rfloor) \bmod M, & \text{for } r = 1, c = 1, \\ (L_{In_r, c, c} + E_{In_{r-1}, N} + \lfloor 2^{32} * P_{In_r, c, c} \rfloor) \bmod M, & \text{for } r = 2 \sim N, c = 1, \\ (L_{In_r, c, c} + E_{In_{r, c-1}, c-1} + \lfloor 2^{32} * P_{In_r, c, c} \rfloor) \bmod M, & \text{for } r = 1 \sim N, c = 2 \sim N. \end{cases} \quad (14)$$

By using the chaotic sequence used for diffusion, an index matrix is generated. Here, L is the final matrix obtained after permutation and P is the chaotic matrix. It is the index matrix obtained by sorting the elements of P and $M = 256$ for 8-bit representation of pixels.

3.1.5 Frame selection

After diffusion, the frames are completely encrypted and are ready to be joined together. To make the encryption even better, the frames are not joined sequentially. The encrypted frames are jumbled using a frame selection key and then are merged to form an encrypted video. This encrypted video seems meaningless to any unauthorized user without the keys and hence can be transmitted securely.

3.2 Decryption

The steps followed for decrypting an encrypted video are shown in Figure 6. Decryption is possible only if all the keys used for encryption and the encrypted video are available. Even a slight change in any of the keys does not produce the original video back.

3.2.1 Frame selection

While generating the encrypted video, a frame selection key is used to jumble the frames to ensure more randomness. The same key is used to place encrypted frames in their correct position before starting the decryption process. Once all the frames are shuffled back to their original places, decryption is done iteratively on each of them.

3.2.2 Antisubstitution

Each encrypted frame is split into three 2D matrices that represent its R, G, and B components as shown in Figure 5. The next step is to revert the changes done by the random order substitution during the diffusion process. It is done by using the following equations:

$$L_{I_r, c, c} = \begin{cases} (E_{I_r, c, c} - E_{I_r, c, c-1} - 2^{32} * P_{I_r, c, c}) \bmod M, & \text{for } r = 1 \sim N, c = 2 \sim N, \\ (E_{I_r, c, c} - E_{I_{r-1}, N} - 2^{32} * P_{I_r, c, c}) \bmod M, & \text{for } r = 2 \sim N, c = 1, \\ (E_{I_r, c, c} - L_{I_{M, N}, N} - 2^{32} * P_{I_r, c, c}) \bmod M, & \text{for } r = 1, c = 1. \end{cases} \quad (15)$$

3.2.3 Rotation

The original image is rotated by 90° in the anticlockwise direction during encryption. The association among adjacent pixels is reduced by using this rotation step. To retrieve the actual image, the encrypted image is rotated by 270° in the counterclockwise direction.

3.2.4 De-permutation

By using the same chaos-based encryption keys used during scrambling, descrambling is done to restore all the pixels to their original positions. This can be considered as the exact opposite process of scrambling.

4 Implementation

This section focuses on the experimental setup including software and hardware requirements for implementing

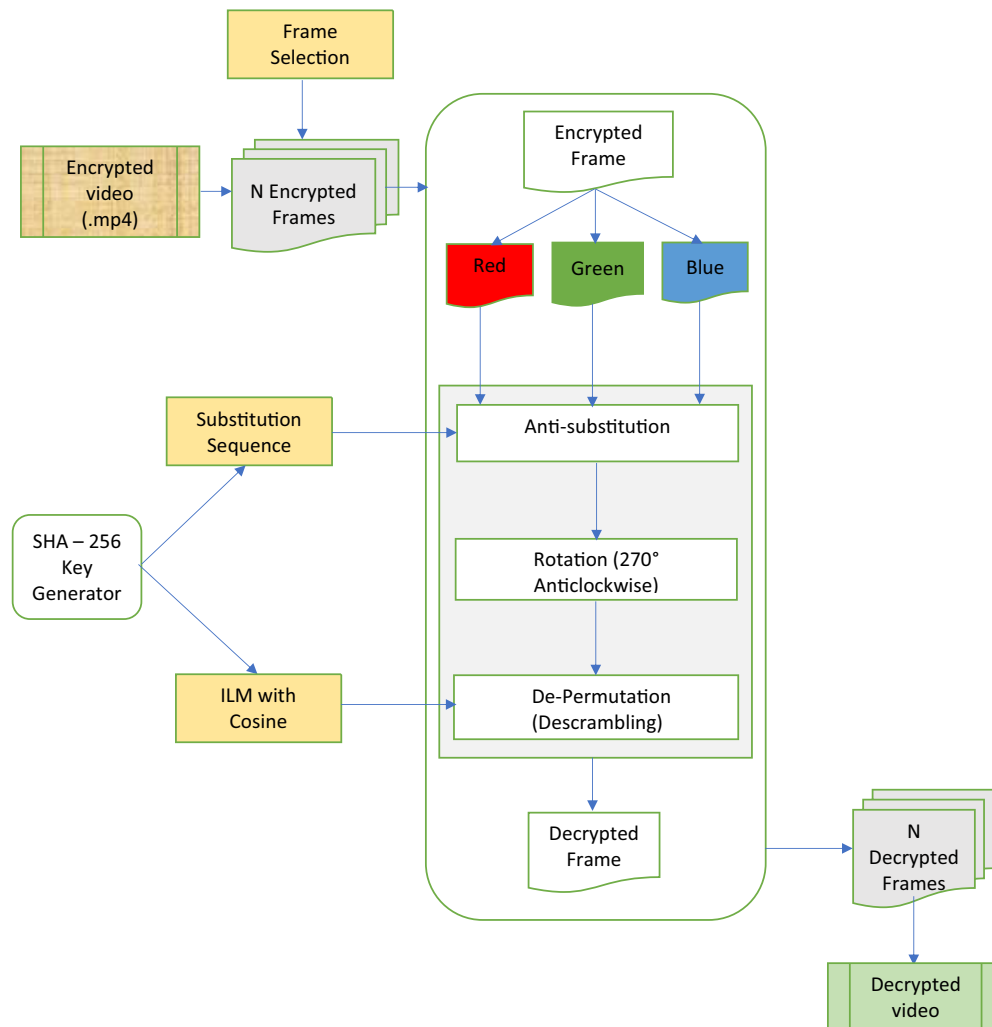


Figure 6: Decryption process of the proposed approach.

the approach proposed in this article. The properties of a sample video taken as input for testing are also explained below along with all the implementation details.

4.1 Experimental setup

The software requirements to implement and test the proposed approach include Python 2.7, version 2.7.17 used on Atom 1.45.0. Libraries like Pillow, OpenCV, NumPy, Matplotlib, and scikit-image are added externally. System configurations are Windows 10, Intel(R) Core (TM) i-5-6200U CPU clocked at 2.30 GHz 2.40 GHz, 8GB RAM, and 64-bit operating system.

The videos for the experiment are taken from ref. [38]. There are four videos with their properties presented in Table 1.

4.2 Encryption

The encryption procedure followed in this article is chaos based. The implementation details explained for encryption and decryption are based on the video titled Flamingo.mp4 from source [38].

Table 1: Properties of various videos used for encryption and decryption

| Video name | Frame per second (FPS) | Length of video (s) | No. of frames | Dimension of frames (Pixels) |
|--------------|------------------------|---------------------|---------------|------------------------------|
| Flamingo.mp4 | 25.0 | 13 | 328 | 352 × 192 |
| Train.mp4 | 25.0 | 10 | 262 | 352 × 192 |
| Rhino.mp4 | 15.0 | 7 | 114 | 320 × 240 |
| Viptrain.mp4 | 30.0 | 20 | 626 | 360 × 240 |

4.2.1 Frame generation

The input video is partitioned into N number of frames, where N is defined by equation (16). N is equivalent to 328 frames for the given input video. The frames are stored in the jpg format, because in this image format, the encryption procedure works faster. Function 1 shows the pseudocode to split the video into frames.

$$N = \text{FPS} * (\text{length of video}). \quad (16)$$

Function 1: Split mp4 video into 2D frames

Input: Original video (n, m) mp4 format, image format (jpg or png)

Output: N ($n \times m$) 2D frames

Pseudocode:

WHILE(True) **DO:**

frame \leftarrow Capture the current frame

Write frame in memory

INCR (currentFrame)

END WHILE

4.2.2 Color frame Input

As the input video is colored, each 2D color frame, which is in jpg format, is converted to a 2D RGB matrix. The RGB matrix is of dimension $3 \times (n \times m)$, where n is the height, m is the width of the image, and each row corresponds to red (R), green (G), and blue (B) colors, respectively. This step is performed to obtain the pixel image for further processing. Function 2 shows the pseudocode to convert the frame into a 2D RGB matrix.

Function 2: Convert a 2D color image to 2D RGB matrix

Input: Original frame (n, m)

Output: 2D IMG_MATRIX ($3 \times (n \times m)$)

Pseudocode:

FOR $x \leftarrow (0, n)$ **DO:**

FOR $y \leftarrow (0, m)$ **DO:**

$r, g, b \leftarrow \text{frame}_{y, x}$

$\text{IMG_MATRIX}[:, x * m + y] \leftarrow [r, g, b]$

END FOR

END FOR

4.2.3 Key generation

ILM chaotic sequence is generated for three-dimensional encryption for R, G, and B components of each frame. Here, the length of the given key is 360. Function 3 shows the pseudocode of secret hash function-256 for seed generation.

Function 3: SHA-256 for 3D seed generation

Input: Length of key (L)

Output: 3D seeds of length L

Pseudocode:

$\text{SECRET_KEY} \leftarrow \text{UNIFORM_RANDOM}(L) // \text{within range } [0, 1]$

$\text{KEY_MATRIX} \leftarrow \text{hash}(\text{SECRET_KEY}, [\text{SHA-256, binary Mode, double}])$

FOR $l \leftarrow (0, L/2)$ **DO:**

$A \leftarrow A \text{ xor } \text{KEY_MATRIX}(l)$

END FOR

FOR gap $\leftarrow (L/2, L)$ **DO:**

$\text{SUM} \leftarrow \text{SUM} + \text{KEY_MATRIX}(\text{gap})$

END FOR

$3\text{D_SEED} \leftarrow A + \text{SUM}$

$3\text{D_SEED} \leftarrow 3\text{D_SEED}/2^{12}$

4.2.4 ILM-cosine sequence generation

Secret key or seeds generated in the previous steps are used to generate a $3 \times (4 \times b \times b)$ ILM-cosine transformation-based chaotic sequences, where b is the block size given in equation (13). Function 4 gives the pseudocode to generate the ILM-cosine sequence, and Table 2 presents a sample ILM-cosine sequence for an 8×8 frame.

Function 4: Generate ILM-Cosine sequence from the secret key

Input: No. of pixels ($4 \times b \times b$), secret key (S)

Output: A $3 \times (4 \times b \times b)$ ILM-cosine chaotic sequence

Pseudocode:

FOR $c \leftarrow (0, 4 \times b \times b)$ **DO:**

$\text{ILM}_{0, c} \leftarrow ((a_1 * S_2 * (1 - S_1)) + S_3) \bmod 1$

$\text{ILM}_{1, c} \leftarrow ((a_1 * S_2) + (S_3 * 1/(1 + \text{ILM}_{0, c}^2))) \bmod 1$

$\text{ILM}_{2, c} \leftarrow (b_1 * (\text{ILM}_{0, c} + \text{ILM}_{1, c} + b_2) * \text{SIN}(S_3)) \bmod 1$

Table 2: ILM-cosine based chaotic sequence for 8×8 frame

| | | | | | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0.679 | 0.939 | 0.877 | 0.120 | 0.234 | 0.077 | 0.017 | 0.618 | 0.336 | 0.584 | 0.756 | 0.825 | 0.883 | 0.017 | 0.753 | 0.004 |
| 0.278 | 0.576 | 0.248 | 0.892 | 0.369 | 0.673 | 0.794 | 0.921 | 0.386 | 0.821 | 0.797 | 0.576 | 0.276 | 0.079 | 0.704 | 0.418 |
| 0.894 | 0.132 | 0.012 | 0.127 | 0.895 | 0.006 | 0.135 | 0.807 | 0.948 | 0.580 | 0.547 | 0.312 | 0.665 | 0.256 | 0.538 | 0.109 |

```

S ← ILM
END FOR
RETURN COS(PI*ILM)

```

4.2.5 Extracting gray components

The 2D RGB matrix image is divided into its corresponding 2D gray components. The gray components of a frame are encrypted individually with a separate ILM sequence for each component. This is done to encrypt the frame from all three dimensions separately. Function 5 shows the pseudocode for extracting the 2D gray components. Figure 7a shows the gray component (red) of dimensions 352×192 from the sample video.

Function 5: Get the 2D gray component from the 2D RGB image

Input: Row, Col, Entire row from RGB matrix that corresponds to the gray component (Data)// $1 \times (n * m)$

Output: 2D GRAY_IMAGE ($n \times m$)

Pseudocode:

GRAY_IMAGE ← create a new image of dimensions (col, row)

```

GRAY_IMAGE ← PUT(Data)
RETURN GRAY_IMAGE

```

4.2.6 Permutation

Permutation is performed using the ILM-cosine transformation-based chaotic sequence generated and is the first step in chaos encryption. Only a fraction of the image is de-correlated by shuffling its pixels. Function 6 shows the pseudocode for scrambling the image, and Figure 7b shows the scrambled gray component (red) of dimensions 352×192 from the sample video.

Function 6: Scramble or permute the pixels to reduce the correlation

Input: GRAY_COMPONENT ($n \times m$), ILM_CSEQ, BLOCK_SIZE(b)

Output: SCRAMBLED_IMG ($n \times m$)

Pseudocode:

CALCULATE Size of the matrix to be scrambled $B \leftarrow b * b$

$P \leftarrow \text{ILM_CSEQ}(0:B)$

$Q \leftarrow \text{ILM_CSEQ}(B:2B)$

$R \leftarrow \text{ILM_CSEQ}(2B:3B)$

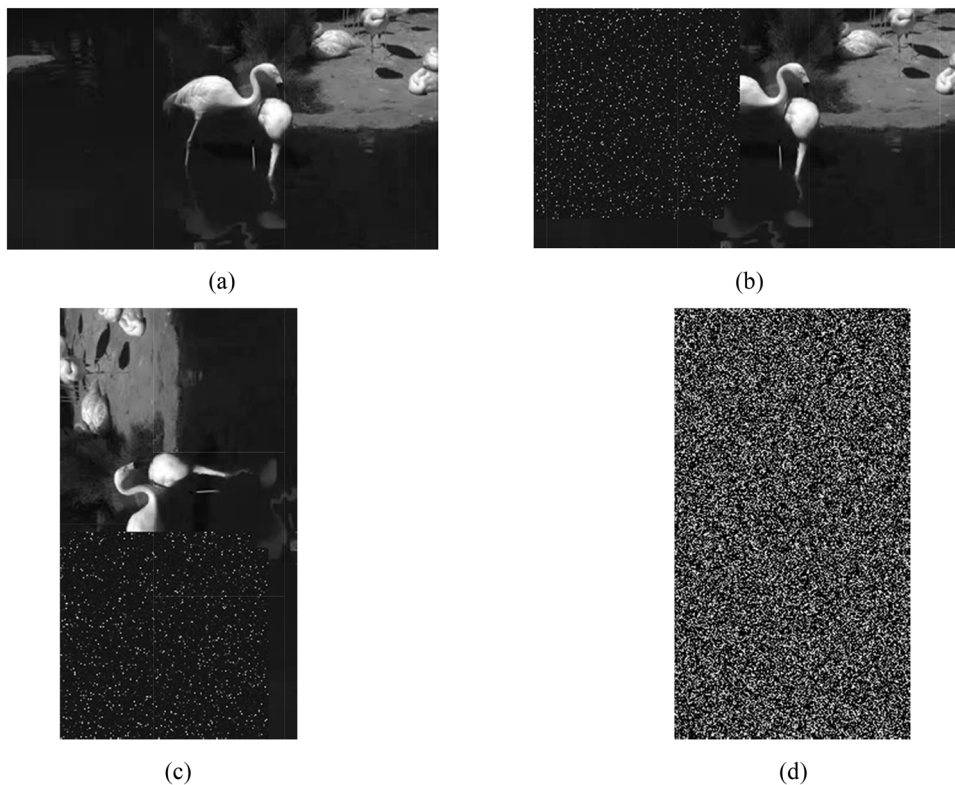


Figure 7: Red component of a frame during encryption process in each step. (a) Original image, (b) scrambled image, (c) rotated image, and (d) encrypted image after substitution.

```

S ← ILM_CSEQ (3B:4B)
In P ← GET_INDEX_SEQ(P) //sort enumerated P matrix
and return P [0]
In Q ← GET_INDEX_SEQ(Q)
In R ← GET_INDEX_SEQ(R)
In S ← GET_INDEX_SEQ(S)
col, row ← SHAPE of GRAY_COMPONENT
SCRAMBLED_IMG ← COPY GRAY_COMPONENT
FOR y ← (1, B + 1) DO:
  FOR x ← (1, B+1) DO:
    c ← (x + In Qy-1 - 1) mod (B + 1)
    d ← (x + In Sy-1 - 1) mod (B + 1)
    Lx, y ← In Pc-1
    Mx, y ← In Rd-1
  END FOR
END FOR
FOR x ← (1, B + 1) DO:
  FOR y ← (1, B + 1) DO:
    i ← Lx, y
    j ← Mi, y
    c1 ← (i - 1)/b
    d1 ← (i - 1) mod b
    c2 ← (j - 1)/b + 1
    d2 ← (j - 1) mod b + 1
    r ← c1*b + c2
    c ← d1*b + d2
    SCRAMBLED_IMGr, c ← imgx, y
  END FOR
END FOR
RETURN SCRAMBLED_IMG

```

4.2.7 Image rotation

Figure 7b shows a scrambled frame. To ensure complete shuffling of pixels from their original positions, the rotation of the frame is done in the anticlockwise direction by the angle of 90 degrees. The rot90 function is used for this purpose. Figure 7c displays the rotated gray component (R) of dimensions 352×192 from the sample video.

4.2.8 Diffusion

Random order substitution is performed using a substitution sequence, which is uniform and random. Function 7 shows the pseudocode for random order substitution, and Figure 7d shows an encrypted 2D gray component (R) of dimensions 352×192 after substitution. After substitution, all the three encrypted gray components are stacked together to obtain the

encrypted 2D RGB image matrix, which is also the final encrypted frame. This encryption process is repeated N times for each frame.

Function 7: Random Order Substitution

Input: GRAY_COMPONENT ($m \times n$)//rotated image, SUB_CSEQ, SIZE

Output: ENCRYPTED_IMG ($m \times n$)

Pseudocode:

GENERATE matrix A and B ← SIZE

A ← ROTATE (Clockwise, 90 degrees, SUB_CSEQ)

In A ← GET_INDEX_SEQ(A)//sort enumerated A matrix and return A [0]

B ← ROTATE (Clockwise, 90 degrees, InA)

M ← 256

FOR r ← (0, row) **DO**:

FOR c ← (0, col) **DO**:

IF r == 0 and c == 0 **THEN**:

 ENCRYPTED_IMG_{Br, c, c} ←

(GRAY_COMPONENT_{Br, c, c} + GRAY_COMPONENT_{Brow-1, col-1, col-1} + $2^{32} * A_{Br, c, c}$) mod M

ELSE IF c == 0 **THEN**:

 ENCRYPTED_IMG_{Br, c, c} ←

(GRAY_COMPONENT_{Br, c, c} + ENCRYPTED_IMG_{Br-1, col-1, col-1} + $2^{32} * A_{Br, c, c}$) mod M

ELSE:

 ENCRYPTED_IMG_{Br, c, c} ←

(GRAY_COMPONENT_{Br, c, c} + ENCRYPTED_IMG_{Br, c-1, j-1} + $2^{32} * A_{Br, c, c}$) mod M

END IF

END FOR

END FOR

RETURN ENCRYPTED_IMG

4.2.9 Frame selection

A random order sequence within the range N , number of frames, is generated and is called frame selection (FS) sequence. The encrypted frames are joined to form an encrypted video in mp4 format according to this FS sequence. Functions 8 and 9 show the pseudocode for generating FS sequence and joining the frames to form a video, respectively.

Function 8: Generate Frame Selection sequence

Input: No. of frames (N)

Output: 1D FS sequence(1xN)

Pseudocode:

```

FOR  $f \leftarrow (0, N)$  DO:
   $r \leftarrow \text{RANDOM\_INT}(1, N)$ 
  IF  $r$  not in FS THEN:
    FS []  $\leftarrow$  Append  $r$ 
  ELSE:
    WHILE  $r$  is less than  $N$  DO:
      INCR( $r$ )
      IF  $r$  more than  $N$  THEN:  $r \leftarrow 1$  END IF
      IF  $r$  not in FS THEN:
        FS []  $\leftarrow$  Append  $r$ 
        BREAK
      END IF
    END WHILE
  END IF
RETURN FS

```

Function 9: Join Frames to get the video

Input: No. of frames(N), FPS

Output: VIDEO_MP4_FPS

Pseudocode

```

FS  $\leftarrow$  FS_SEQUENCE( $N$ )
frame  $\leftarrow$  Read First frame
 $h, w \leftarrow \text{SHAPE}(\text{frame})$ 
OUTPUT_FORMAT  $\leftarrow$  SET the output video format
size  $\leftarrow (w, h)$ 
SET VIDEO_WRITER using FPS, OUTPUT_FORMAT,
size, OUTPUT_PATH
FOR  $f \leftarrow (0, N)$  DO:
  IMAGE_PATH  $\leftarrow$  SET frame path
  Img  $\leftarrow$  Read (IMAGE_PATH)
  Write Img using VIDEO_WRITER
END FOR

```

4.3 Decryption

Input for the decryption process is the encrypted video file in mp4 format, and the output is decrypted video file in mp4 format, which is similar to the original input video. To test the quality of decryption, MSE and PSNR tests are performed, the results of which are analyzed in the next section.

4.3.1 Frame generation

Encrypted video is in mp4 format and to decrypt the video, it is partitioned into N frames, where N is derived from equation (16). The procedure to partition the video is similar to that given in the encryption process and is shown in Function 1. The frames are saved in the memory in .png format.

4.3.2 Rearrange frames

The next step is to rearrange the frames in their actual order using the same FS sequence. The process is simply renaming the frames in order of their occurrence according to the FS sequence. After this process, frames are in their actual order and are ready to be decrypted.



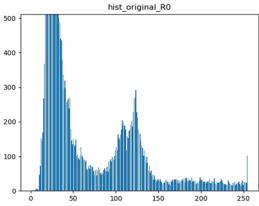
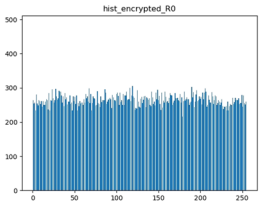


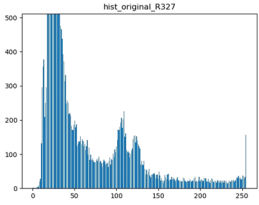
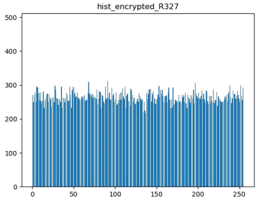

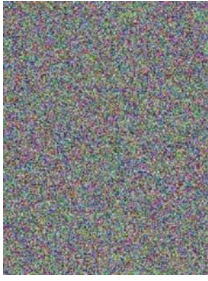
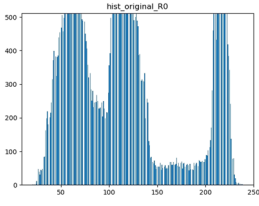
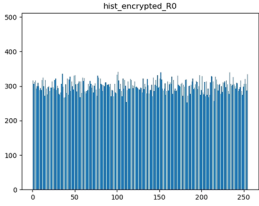

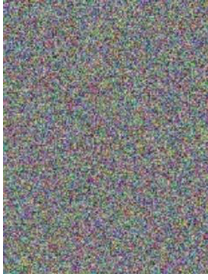
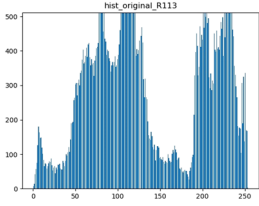
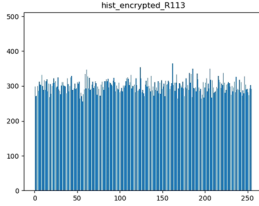


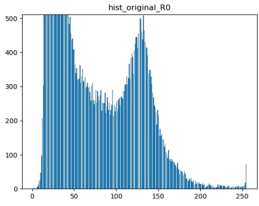
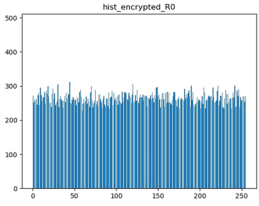

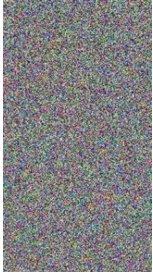
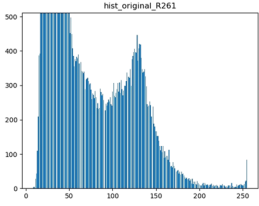
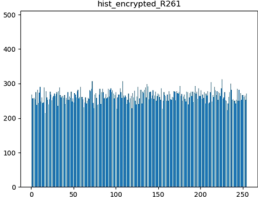
4.3.3 Extracting gray components

The .png format 2D image file is initially converted to its corresponding 2D RGB image matrix. This step is

Table 3: UACI, NPCR, CC, and entropy analysis

| Video (mp4) | Component | UACI | NPCR | CC _h | CC _v | CC _d | Entropy |
|-------------|-----------|----------|----------|-----------------|-----------------|-----------------|---------|
| Flamingo | Red | 33.76992 | 99.60848 | -0.001483 | +0.004846 | +0.000177 | 7.99719 |
| | Green | 35.34373 | 99.62372 | -0.012479 | +0.025968 | -0.005657 | 7.99731 |
| | Blue | 32.72501 | 99.61854 | +0.006315 | -0.003929 | -0.001170 | 7.99732 |
| Rhino | Red | 33.23251 | 99.61549 | +0.009020 | -0.014905 | +0.000101 | 7.99757 |
| | Green | 33.69045 | 99.60130 | +0.013223 | +0.004301 | -0.005752 | 7.99762 |
| | Blue | 32.77237 | 99.61276 | +0.006237 | -0.007078 | -0.001706 | 7.99746 |
| Train | Red | 33.79639 | 99.61573 | -0.014568 | +0.003141 | +0.000397 | 7.99726 |
| | Green | 32.90723 | 99.62328 | +0.016226 | +0.020616 | -0.000320 | 7.99727 |
| | Blue | 37.66040 | 99.60256 | -0.001335 | -0.006978 | -0.000918 | 7.99723 |
| Viptrain | Red | 32.23780 | 99.60798 | -0.003143 | +0.005592 | -0.006909 | 7.99791 |
| | Green | 31.98262 | 99.61377 | -0.001178 | -0.003753 | +0.002703 | 7.99774 |
| | Blue | 32.38443 | 99.61990 | -0.007492 | +0.015163 | +0.010275 | 7.99788 |

Table 4: Histogram analysis

| Video (mp4) | Original frame | Encrypted frame | Original frame Histogram | Encrypted frame Histogram |
|-------------|---|---|--|---|
| Flamingo |  |  |  |  |
| |  |  |  |  |
| Rhino |  |  |  |  |
| |  |  |  |  |
| Train |  |  |  |  |
| |  |  |  |  |

(Continued)

Table 4: Continued



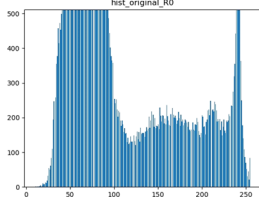
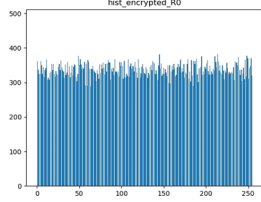

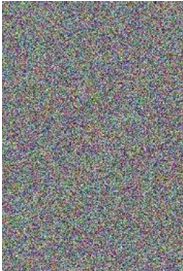
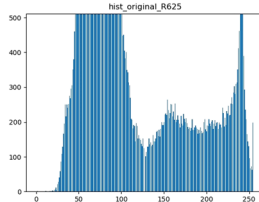
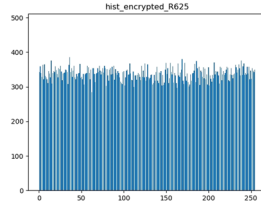
| Video (mp4) | Original frame | Encrypted frame | Original frame Histogram | Encrypted frame Histogram |
|-------------|---|---|--|---|
| Viptrain |  |  |  |  |
| |  |  |  |  |
| | | | | |
| | | | | |

Table 5: PSNR and MSE

| Video (mp4) | Component | PSNR | MSE |
|-------------|------------|-------------------|-----------------------|
| Flamingo | Full frame | 36.6755606209867 | 15.148983651620369933 |
| | Red | 35.94202076570667 | 17.697918639520202067 |
| | Green | 37.237718687633 | 13.016120186237373633 |
| | Blue | 36.902444685967 | 14.732912129103535067 |
| Rhino | Full frame | 31.5629403311267 | 49.8130211226851868 |
| | Red | 31.4234534274933 | 51.312109375 |
| | Green | 31.39083511636 | 50.851566840278 |
| | Blue | 31.7684474798 | 47.27538715279 |
| Train | Full frame | 34.6309128541933 | 23.8425902909301348 |
| | Red | 34.6470446036933 | 24.447270557133839067 |
| | Green | 33.7075822148267 | 32.875726010101009533 |
| | Blue | 37.2621035020533 | 14.204774305555555067 |
| Viptrain | Full frame | 32.0135713627867 | 40.992180298353909067 |
| | Red | 31.1408029721133 | 50.619496913580246133 |
| | Green | 32.328658491467 | 38.004512345679013 |
| | Blue | 32.806820655233 | 34.352531635802468933 |

performed to get the 2D pixel image for further processing. The pseudocode for the same is shown in Function 2. The frames are encrypted in three dimensions (R, G, and B) individually using a separate ILM sequence for every gray component, and hence, to decrypt the frame, all three gray components are extracted from the 2D RGB image matrix according to Function 5.

4.3.4 Reversal of diffusion

Reversal of diffusion or antisubstitution is to replace the pixels by substituting them with the original values using the same substitution sequence used during encryption. Function 10 shows the pseudocode for antisubstitution using random order substitution.

Function 10: Substitute pixels back to their original pixel value

Input: ENCRYPTED_IMG ($m \times n$), SUB_CSEQ, SIZE

Output: ASUB_IMG ($m \times n$)

Pseudocode:

```

GENERATE matrix A and B  $\leftarrow$  SIZE
A  $\leftarrow$  ROTATE (Clockwise,  $90^\circ$ , SUB_CSEQ)
InA  $\leftarrow$  GET_INDEX_SEQ(A)//sort enumerated A matrix
and return A [0]
B  $\leftarrow$  ROTATE (Clockwise,  $90^\circ$ , In A)
M  $\leftarrow$  256
  FOR r  $\leftarrow$  (0, row) DO:
    FOR c  $\leftarrow$  (0, col) DO:
      ASUB_IMGBr, c, c  $\leftarrow$  (M + ENCRYPTED_IMGBr, c, c -
- ENCRYPTED_IMGBr, c-1, c-1 -  $2^{32} \cdot A_{Br, c, c}$ ) mod M
    END FOR
  END FOR
c  $\leftarrow$  0
  FOR r  $\leftarrow$  (1, col) DO:
    ASUB_IMGBr, c, c  $\leftarrow$  (M + ENCRYPTED_IMGBr, c, c -
ENCRYPTED_IMGBr-1, col-1, col-1 -  $2^{32} \cdot A_{Br, c, c}$ ) mod M
  END FOR
i  $\leftarrow$  0
ASUB_IMGBr, c, c  $\leftarrow$  (M + ENCRYPTED_IMGBr, c, c -
ASUB_IMGBrow-1, col-1, col-1 -  $2^{32} \cdot A_{Br, c, c}$ ) mod M
RETURN ASUB_IMG

```

4.3.5 Image rotation

During decryption, the frame is rotated 270° counter-clockwise. The rot90 function of numpy library in python is used three times for this purpose.

4.3.6 Descrambling

The inverse of permutation called descrambling is performed as the final step of decryption to get the original frame. Function 11 shows the pseudocode for descrambling the image. After substitution, all the three decrypted gray components are stacked together to obtain the decrypted 2D RGB image matrix, which is also the final decrypted frame. This decryption process is repeated N times for each frame, and all the decrypted frames are joined together to generate mp4 decrypted video as shown in Function 9.

Function 11: Correlate the pixel back to their original position

Input: SCRAMBLED_IMG ($n \times m$), ILM_CSEQ, BLOCK_SIZE(b)

Output: ORIG_GRAY_IMG ($n \times m$)

Pseudocode:

```















CALCULATE Size of the matrix to be descrambled B  $\leftarrow$  b*b
P  $\leftarrow$  ILM_CSEQ (0: B)
Q  $\leftarrow$  ILM_CSEQ (B:2B)
R  $\leftarrow$  ILM_CSEQ (2B:3B)
S  $\leftarrow$  ILM_CSEQ (3B:4B)
In P  $\leftarrow$  GET_INDEX_SEQ(P)//sort enumerated P matrix
and return P [0]
In Q  $\leftarrow$  GET_INDEX_SEQ(Q)
In R  $\leftarrow$  GET_INDEX_SEQ(R)
In S  $\leftarrow$  GET_INDEX_SEQ(S)
col, row  $\leftarrow$  SHAPE of GRAY_COMPONENT
ORIG_GRAY_IMG  $\leftarrow$  COPY SCRAMBLED_IMG
FOR y  $\leftarrow$  (1, B + 1) DO:
  FOR x  $\leftarrow$  (1, B + 1) DO:
    c  $\leftarrow$  (x + In Qy-1 - 1) mod (B + 1)
    d  $\leftarrow$  (x + In Sy-1 - 1) mod (B + 1)
    Lx, y  $\leftarrow$  In Pc-1
    Mx, y  $\leftarrow$  In Rd-1
  END FOR
END FOR
FOR x  $\leftarrow$  (1, B + 1) DO:
  FOR y  $\leftarrow$  (1, B + 1) DO:
    i  $\leftarrow$  Lx, y
    j  $\leftarrow$  Mi, y
    c1  $\leftarrow$  (i - 1)/b
    d1  $\leftarrow$  (i - 1) mod b
    c2  $\leftarrow$  (j - 1)/b + 1
    d2  $\leftarrow$  (j - 1) mod b + 1
    r  $\leftarrow$  c1*b + c2
    c  $\leftarrow$  d1*b + d2
    ORIG_GRAY_IMGy  $\leftarrow$  SCRAMBLED_IMGr, c
  END FOR
END FOR
RETURN ORIG_GRAY_IMG

```

Table 6: Encryption Speed (FPS)

| Video (mp4) | Frame size (Pixels) | Time per frame (in seconds) | |
|-------------|---------------------|-----------------------------|------------|
| | | Encryption | Decryption |
| Flamingo | 8 × 8 | 0.00450 | 0.00001 |
| | 128 × 128 | 1.19019 | 1.00000 |
| | 352 × 192 | 3.76710 | 6.00000 |
| Rhino | 8 × 8 | 0.00580 | 0.00001 |
| | 128 × 128 | 1.24229 | 1.06667 |
| | 320 × 240 | 4.02550 | 5.30000 |
| Train | 8 × 8 | 0.00420 | 0.00001 |
| | 128 × 128 | 1.16879 | 1.00000 |
| | 352 × 192 | 4.49179 | 5.00000 |
| Viptrain | 8 × 8 | 0.01139 | 0.00001 |
| | 128 × 128 | 1.20070 | 1.46667 |
| | 360 × 240 | 5.49750 | 5.20000 |

Table 7: Original and decrypted frames

| Video | Original frame | Decrypted frame |
|----------|---|---|
| Flamingo |  |  |
| |  |  |
| Rhino |  |  |
| |  |  |
| Train |  |  |
| |  |  |
| Viptrain |  |  |

(Continued)

Table 7: Continued

| Video | Original frame | Decrypted frame |
|-------|---|---|
| |  |  |

5 Results and analysis

To compare the encryption efficiency of the proposed method with existing methods, some parameters were used. They are described as follows.

5.1 Differential attacks

Differential attacks [39] are the most common kind of attacks performed on block ciphers. Two parameters, NPCR and UACI, where the former is described as the number of pixel change rate and the latter is defined as unified averaged change in intensity, were used to evaluate the performance of encryption algorithms against these attacks. UACI is calculated using equation (17).

$$UACI = \frac{1}{A \times B} \sum_{x=1}^A \sum_{y=1}^B \frac{|E_1(x, y) - E_2(x, y)|}{255} \times 100\%, \quad (17)$$

where $E(x, y)$ is the pixel value present in the x th row and the y th column of E . E_1 and E_2 denote the encrypted images. According to various experiments done, the ideal value of UACI is 33.

NPCR is the rate of change in the number of pixels in an encrypted image when one pixel is modified in the original image. It focuses on the absolute number of pixels, which change after an attack. It is calculated using the following formula:

$$NPCR = \frac{1}{A \times B} \sum_{x=1}^A \sum_{y=1}^B U(x, y) \times 100\%, \quad (18)$$

where

Table 8: Comparison with existing approaches

| Criteria | Valli and Ganesan [38] | Deshmukh and Kolhe [40] | Ranjith kumar et al. [20] | Proposed approach | | |
|--------------------------------------|---------------------------------|--------------------------------|---------------------------|-------------------|------------|-----------|
| Video | Rhino.mp4 | Foreman.mpeg [40] | Rhino.mp4 | Rhino.mp4 | | |
| Frame size | — | — | — | 128 × 128 | | |
| | | | | Red | Green | Blue |
| NPCR | 99.4518 | — | 99.51 | 99.61549 | 99.60130 | 99.61276 |
| UACI | 33.63 | — | 33.54 | 33.23251 | 33.69045 | 32.77237 |
| CC h | 0.0181 | −0.0112 | 0.0324 | 0.009020 | 0.013223 | 0.006237 |
| CC v | 0.0140 | −0.0813 | 0.0261 | −0.01490 | 0.004301 | −0.00707 |
| CC d | 0.0107 | 0.0009 | 0.0263 | 0.000101 | −0.00575 | −0.00170 |
| Entropy analysis | — | 7.941 | — | 7.99757 | 7.99762 | 7.99746 |
| Histogram | Uniform | — | Uniform | Uniform | Uniform | Uniform |
| MSE | — | — | — | 51.312109 | 50.851566 | 47.275387 |
| PSNR | — | — | — | 31.423453 | 31.3908351 | 31.768447 |
| Key space | Size * 2 ¹²⁸ | 128, 192, 256 | 2 ²¹² | SHA-2 | | |
| Encryption time + key generation (s) | 1.2147 (without key generation) | 1.122 (without key generation) | 2.85878 | 1.24229 | | |
| Decryption time (s) | — | 2.540 | 1.85082 | 1.06667 | | |

$$U(x, y) = \begin{cases} 0, & c1(x, y) \neq c2(x, y) \\ 1, & \text{in other cases.} \end{cases} \quad (19)$$

The ideal value of NPCR is 99. The values obtained for UACI and NPCR by the proposed work are presented in Table 3.

5.2 Correlation coefficient (CC) analysis

CC denotes the association between two adjacent pixels in an image. Its value lies within the range $[-1, 1]$, where 0 is considered the ideal value. The value 0 means that there is no correlation, and the value 1 indicates a high correlation among the pixels. CC is calculated by choosing 1,000 pairs of pixels from the image and forming duplets from them. It is estimated for pixel pairs horizontally (CC_h), vertically (CC_v), and diagonally (CC_d) and can be calculated using the following formulae:

$$c_{i,j} = \frac{\text{cov}(i, j)}{\sqrt{A(i)}\sqrt{A(j)}}, \quad (20)$$

$$\text{where, } \text{cov}(i, j) = \frac{1}{M} \sum_{k=1}^M (x_k - B(x))(y_k - B(y)), \quad (21)$$

$$A(x) = \frac{1}{M} \sum_{k=1}^M (x_k - B(x))^2, \quad (22)$$

$$B(x) = \frac{1}{M} \sum_{k=1}^M x_k. \quad (23)$$

Here, $c_{i,j}$ denotes the CC, M denotes the pixel pairs selected randomly, and i and j denote two adjacent pixels. $A(x)$ and $B(x)$ denote the variance and expectation of x , respectively. Table 3 presents the results of CC for the proposed approach.

5.3 Entropy analysis

Entropy is the degree of randomness of an image. An encrypted image with entropy value 8 is said to have the maximum degree of randomness and hence is resistant to differential attacks. Entropy is calculated using equation (24).

$$E(i) = \sum_{j=0}^{G-1} p(i_j) \frac{1}{p(i_j)}, \quad (24)$$

where E denotes entropy and $G = 2^k$ and $k = 8$ for gray scale image. G denotes the number of states of the frame analyzed. Table 3 presents the entropy analysis for the proposed approach.

5.4 Histogram analysis

This analysis depicts the level of encryption and can be used to determine the strength of the used algorithm against attacks. A uniform pattern in the histogram implies a good encryption scheme, which is difficult to crack. Table 4 presents the histogram analysis of unencrypted and encrypted frames.

5.5 MSE and PSNR analysis

Mean square error (MSE) is the average of the difference of squares of intensity between the encrypted and plain images. It can be calculated using equation (25) [41,42].

Peak signal-to-noise ratio (PSNR) is the measure of change of quality between actual and encrypted images. In the case of image data, the original image acts as the signal, and error is the noise generated because of encryption. A high value of PSNR means that the decrypted image is of good quality. It can be calculated using equation (26) by using the MSE value [43,44].

$$\text{MSE} = \frac{1}{m \times n} \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} [O(x, y) - E(x, y)]^2, \quad (25)$$

$$\text{PSNR} = 10 \log_{10} \frac{(\text{MAX}_I)^2}{\text{MSE}}, \quad (26)$$

where E is the image with noise, O is the original image with dimensions $m \times n$, and MAX_I is the pixel with maximum value in the image. The value of MAX_I is 255 for 8-bit representation of pixels. Table 5 presents the MSE and PSNR test values for the proposed approach.

5.6 Time analysis

An encryption algorithm is considered to be good if it takes less amount of time to process data without compromising the level of security. The major factor that determines the time complexity of the proposed approach is the size of the frames. Images with bigger dimensions take more time for encryption and decryption when compared to smaller images. The analysis for different values of frame size is presented in Table 6.

Table 7 shows the comparison between actual and decrypted frames. Table 8 shows a detailed comparison between the proposed approach and other video encryption models, which already exist. It can be observed that the UACI and NPCR values obtained by the proposed

method are very close to the ideal values mentioned in Section 5.1. The ideal value of CC is 0. From Table 8, it is clear that the CC value of the proposed approach is closer to 0 when compared to other methods under inspection. Similarly, the maximum value of entropy is 8, and the entropy of the proposed method is better and much closer to 8 than other methods.

MSE and PSNR values play an important role to determine the quality of image encryption and decryption respectively. Hence, the values obtained after MSE and PSNR analysis are also populated in the table. Finally, the time taken for encryption and decryption is compared to get an idea of the faster algorithm among those under comparison.

6 Conclusion and future works

On analyzing the data presented in Section 5, it is evident that the proposed approach for video encryption and decryption is more secure and faster than all the existing methods. The keys produced by combining SHA-2 with cosine-based ILM are more uniform, nonlinear, and better. As a result, the obtained values of various testing parameters are very close to the ideal values and indicate that the proposed approach is very favorable for secure video encryption.

In the real world, this method of encryption and decryption can be used to send and receive sensitive medical, military, or any other video data of high importance. Furthermore, this approach can also be integrated with social media to make the data sharing more reliable. In the future, efforts could be made to incorporate audio encryption in the process and achieve better results, within less time.

Conflict of interest: The submitted work does not have any conflict of interest.

Data availability statement: Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

References

- [1] M. Agrawal and P. Mishra, "A comparative survey on symmetric key encryption techniques," *Int. J. Computer Sci. Eng.*, vol. 4, no. 5, p. 877, 2012.
- [2] A. Lindahl, M. Girish, and C. Duvivier, *U.S. Patent Application No. 11/247,955*, 2007.
- [3] S. Lian, *Multimedia content encryption: techniques and applications*, CRC press, 2008.
- [4] L. Qiao and K. Nahrstedt, "A new algorithm for MPEG video encryption," *Proc. of First International Conference on Imaging Science System and Technology*, 1997, July, pp. 21–29.
- [5] C. Shi and B. Bhargava, "A fast MPEG video encryption algorithm," *Proceedings of the sixth ACM international conference on Multimedia*, 1998, September, pp. 81–88.
- [6] C. Shi and B. Bhargava, "An efficient MPEG video encryption algorithm," *Proceedings Seventeenth IEEE Symposium on Reliable Distributed Systems (Cat. No. 98CB36281)*, IEEE, 1998, October, pp. 381–386.
- [7] B. Bhargava, C. Shi, and S. Y. Wang, "MPEG video encryption algorithms," *Multimed. Tools Appl.*, vol. 24, no. 1, pp. 57–79, 2004.
- [8] S. S. Maniccam and N. G. Bourbakis, "Image and video encryption using SCAN patterns," *Pattern Recognit.*, vol. 37, no. 4, pp. 725–737, 2004.
- [9] S. Li, G. Chen, and X. Zheng, "Chaos-based encryption for digital image and video," *Multimedia Encryption and Authentication Techniques and Applications*, 2006, p. 129.
- [10] C. N. Raju, G. Umadevi, K. Srinathan, and C. V. Jawahar, "Fast and secure real-time video encryption," *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, Bhubaneswar, India: IEEE, 2008, December, pp. 257–264.
- [11] S. Lian, "Efficient image or video encryption based on spatio-temporal chaos system," *Chaos, Solitons & Fractals*, vol. 40, no. 5, pp. 2509–2519, 2009.
- [12] P. Deshmukh and V. Kolhe, "Modified AES based algorithm for MPEG video encryption," *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Chennai, India: IEEE, 2014, February, pp. 1–5.
- [13] M. Altaf, A. Ahmad, F. A. Khan, Z. Uddin, and X. Yang, "Computationally efficient selective video encryption with chaos based block cipher," *Multimed. Tools Appl.*, vol. 77, no. 21, pp. 27981–27995, 2018.
- [14] F. Chiaraluce, L. Ciccirelli, E. Gambi, P. Pierleoni, and M. Reginelli, "A new chaotic algorithm for video encryption," *IEEE Trans. Consum. Electron.*, vol. 48, no. 4, pp. 838–844, 2002.
- [15] Y. Li, L. Liang, Z. Su, and J. Jiang, "A new video encryption algorithm for H. 264," *In 2005 5th International Conference on Information Communications & Signal Processing*, Bangkok: IEEE, 2005 December, pp. 1121–1124.
- [16] C. N. Raju, G. Umadevi, K. Srinathan, and C. V. Jawahar, "Fast and Secure Real-Time Video Encryption," *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, Bhubaneswar: IEEE, 2008, pp. 257–264.
- [17] D. M. Dumbere and N. J. Janwe, "Video encryption using AES algorithm," *Second International Conference on Current Trends In Engineering and Technology - ICCTET 2014*, Coimbatore: IEEE, 2014, pp. 332–337.
- [18] M. Preishuber, T. Hütter, S. Katzenbeisser, and A. Uhl, "Depreciating motivation and empirical security analysis of chaos-based image and video encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 9, pp. 2137–2150, 2018.
- [19] W. Wen, R. Tu and K. Wei, "Video frames encryption based on DNA sequences and chaos," *Eleventh International Conference on Digital Image Processing (ICDIP 2019)*, vol. 11179, International Society for Optics and Photonics, 2019, August, p. 111792T.
- [20] R. Ranjith kumar, D. Ganeshkumar, A. Suresh, and K. Manigandan, "A new one Round video encryption scheme based on 1D chaotic maps," *2019 5th International Conference*

- on *Advanced Computing & Communication Systems (ICACCS)*, Coimbatore, India: IEEE, 2019, pp. 439–444.
- [21] G. Ye and X. Huang, “An efficient symmetric image encryption algorithm based on an intertwining logistic map,” *Neurocomputing*, vol. 251, pp. 45–53, 2017.
- [22] Z. Hua, F. Jin, B. Xu, and H. Huang, “2D Logistic-Sine-coupling map for image encryption,” *Signal. Process.*, vol. 149, pp. 148–161, 2018.
- [23] Z. Hua, Y. Zhou, and H. Huang, “Cosine-transform-based chaotic system for image encryption,” *Inf. Sci.*, vol. 480, pp. 403–419, 2019.
- [24] M. R. Guevara and L. Glass, “Phase locking, period doubling bifurcations and chaos in a mathematical model of a periodically driven oscillator: A theory for the entrainment of biological oscillators and the generation of cardiac dysrhythmias,” *J. Math. Biol.*, vol. 14, no. 1. pp. 1–23, 1982.
- [25] X. Lu, D. Clements-Croome, and M. Viljanen, “Integration of chaos theory and mathematical models in building simulation: part I: Literature review,” *Autom. Constr.*, vol. 19, no. 4. pp. 447–451, 2010.
- [26] K. Mischaikow and M. Mrozek, “Chaos in the Lorenz equations: a computer-assisted proof,” *Bull. Am. Math. Soc.*, vol. 32, no. 1. pp. 66–72, 1995.
- [27] J. Fridrich, “Image encryption based on chaotic maps,” *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 2, Orlando, FL, USA: IEEE, 1997, October, pp. 1105–1110.
- [28] S. C. Phatak and S. S. Rao, “Logistic map: A possible random-number generator,” *Phys. Rev. E*, vol. 51, no. 4. p. 3670, 1995.
- [29] J. Fridrich, “Symmetric ciphers based on two-dimensional chaotic maps,” *Int. J. Bifurc. chaos*, vol. 8, no. 6. pp. 1259–1284, 1998.
- [30] P. N. Khade and M. Narnaware, “3D chaotic functions for image encryption,” *Int. J. Computer Sci. Issues (IJCSI)*, vol. 9, no. 3. p. 323, 2012.
- [31] M. Kumar, S. Kumar, R. Budhiraja, M. K. Das, and S. Singh, “Intertwining logistic map and Cellular Automata based color image encryption model,” *2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT)*, New Delhi, India: IEEE, 2016, March, pp. 618–623.
- [32] R. M. Lin and T. Y. Ng, “Secure image encryption based on an ideal new nonlinear discrete dynamical system,” *Math. Probl. Eng.*, p. 2018, 2018.
- [33] X. Wang and D. Xu, “Image encryption using genetic operators and intertwining logistic map,” *Nonlinear Dyn.*, vol. 78, no. 4. pp. 2975–2984, 2014.
- [34] B. K. Nancharla and M. Dua, “An image encryption using intertwining logistic map and enhanced logistic map,” *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India: IEEE, 2020, June, pp. 1309–1314.
- [35] M. Dua, A. Suthar, A. Garg, and V. Garg, “An ILM-cosine transform-based improved approach to image encryption,” *Complex. & Intell. Syst.*, pp. 1–17, 2020.
- [36] H. Gilbert and H. Handschuh, “Security analysis of SHA-256 and sisters,” *International Workshop on Selected Areas in Cryptography*, Berlin, Heidelberg, Springer, 2003, August, pp. 175–193.
- [37] S. Gueron, S. Johnson, and J. Walker, “SHA-512/256,” *2011 Eighth International Conference on Information Technology: New Generations*, Las Vegas, NV, USA: IEEE, 2011, April, pp. 354–358.
- [38] D. Valli and K. Ganesan, “Chaos based video encryption using maps and Ikeda time delay system,” *European Phys. J. plus*, vol. 132, p. 542, 2017, doi: 10.1140/epjp/i2017-11819-7
- [39] Y. Wu, J. P. Noonan, and S. Agaian, “NPCR and UACI randomness tests for image encryption,” *Cyber J.: Multidiscip. journals Sci. Technol. J. Sel. Areas Telecommun. (JSAT)*, vol. 1, no. 2. pp. 31–38, 2011.
- [40] P. Deshmukh and V. Kolhe, “Modified AES based algorithm for MPEG video encryption,” *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Chennai: IEEE, 2014, pp. 1–5. doi: 10.1109/ICICES.2014.7033928.
- [41] M. Dua, A. Wesanekar, V. Gupta, M. Bhola, and S. Dua, “Differential evolution optimization of intertwining logistic map-DNA based image encryption technique,” *J. Ambient. Intell. Humanized Comput.*, vol. 11, no. 9, pp. 1–16, 2019.
- [42] A. Bisht, M. Dua, and S. Dua, “A novel approach to encrypt multiple images using multiple chaotic maps and chaotic discrete fractional random transform,” *J. Ambient. Intell. Humanized Comput.*, vol. 10, no. 9. pp. 3519–3531, 2019.
- [43] M. Dua, A. Wesanekar, V. Gupta, M. Bhola, and S. Dua, “Color image Encryption using synchronous CML-DNA and weighted bi-objective genetic algorithm,” *Proceedings of the 3rd International Conference on Big Data and Internet of Things*, 2019, August, pp. 121–125.
- [44] A. Bisht, M. Dua, S. Dua, and P. Jaroli, “A color image encryption technique based on bit-level permutation and alternate logistic maps,” *J. Intell. Syst.*, vol. 29, no. 1. pp. 1246–1260, 2019.