J. C. Bose Journal of Engineering, Technology and Sciences
Vol. 1, Issue No. 1, March 2025

Performance Comparison of Traditional Block and Stream Ciphers in Video Encryption

Pratham Jain, Sudhanshu Chaudhary and Samyak Jain*

Department of Computer Engineering, J. C. Bose University of Science and Technology, YMCA, Faridabad 121006, Haryana, India

Abstract: In modern times, ensuring confidentiality is the utmost thing that needs to be done for all domains. Data security is paramount to protect sensitive information from unauthorized access and cyber threats. This paper focuses on the confidentiality of video data. Encryption is a widely accepted solution to protect videos from attacks and unauthorized access. There are different cryptographic algorithms, which are further classified into block and stream ciphers. In block cipher, a fixed-size data block is taken at a time and encrypted. Encryption occurs in different rounds where the number of rounds depends upon the key size, whereas in stream ciphers, a byte of data is encrypted at a time. This paper presents a comprehensive performance comparison of block and stream ciphers, block ciphers including Advanced Encryption Standard (AES), Data Encryption Standard (DES), Rivest Cipher 5(RC5), RC6 algorithms and stream ciphers including RC4 algorithm. In this paper, the performance of these algorithms is evaluated based on Quantitative, Qualitative and Robustness analysis. Testing is performed on three distinct video datasets for experimentation. The study concludes that block ciphers offer superior security around 5-15% when compared with stream ciphers. These findings underscore the critical role of choosing appropriate encryption methods to effectively protect video data from evolving cyber threats.

Keywords: *Block cipher, Encryption, Qualitative analysis, Quantitative analysis, Robustness analysis, Stream cipher.*

1. Introduction

In today's digital era, where everything has become digital, security is necessary to safeguard sensitive information. Unauthorized access to critical data, like personal details, financial records, or national security secrets, can have devastating consequences

Security ensures data confidentiality, protecting sensitive information from unauthorized access or interception. Encryption serves as a cornerstone of digital security, ensuring data confidentiality by rendering it such that it is unreadable to the naked eye.

In the last measured period, the number of data breaches in the United States amounted to 3,205 cases, with over 353 million records exposed. This shows that attacks are a significant concern in security. From Figure 1, the statistics confirm the defined facts (Petrosyan, 2024). These figures are significant and need rigorous efforts to be safeguarded.

Encryption helps maintain data privacy, integrity, and authenticity, safeguarding individuals, organizations and governments from cyber threats and breaches. Thus, encryption and decryption are essential for maintaining trust, compliance with privacy regulations, and securing sensitive information in the digital age.

*Corresponding author
Email address: samyakjainsamy2004@gmail.com

Received 19 July 2024; Accepted 17 September 2024
Published online 8 March 2025

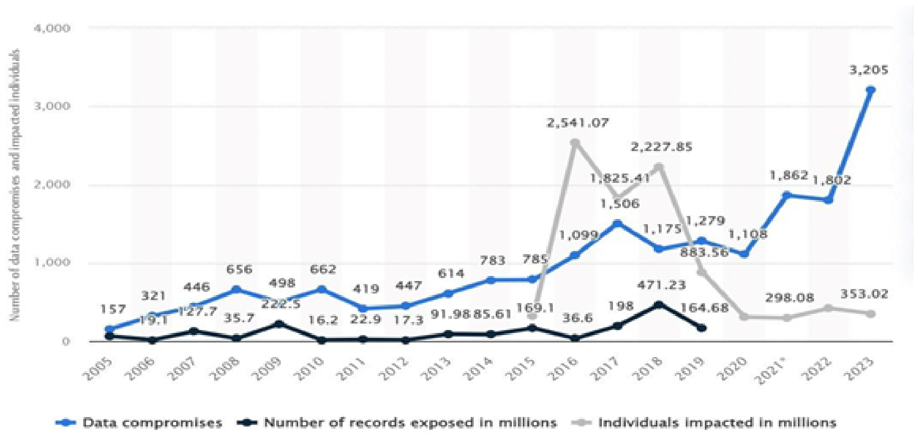


Figure 1: Number of data breaches in US.

2. Various Types of Mechanisms

Cryptographic mechanisms can be classified on a diverse basis. This paper focuses on the two broad categories, block ciphers and stream ciphers, as shown in Figure 2. Block ciphers encrypt data in fixed-size blocks, providing high security and attack resistance. Examples include AES, DES, RC5, and RC6. Stream ciphers, like RC4, encrypt data continuously like a stream, offering speed and efficiency, especially for real-time data encryption (Sahu, 2023).

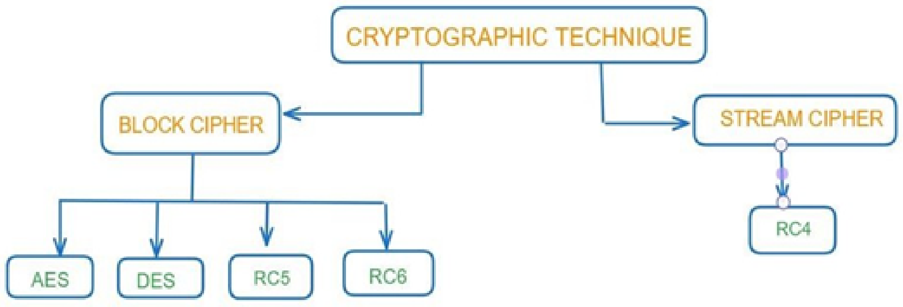


Figure 2: Cryptographic Techniques.

2.1. Block Cipher

In these algorithms, data encryption and decryption are done in the form of blocks of data. In it, the plain text is divided into blocks and then fed into the cipher system to produce blocks of cipher text. The basic form of block cipher is ECB (Electronic Codebook Mode) where data blocks are encrypted directly to generate their corresponding cipher blocks (Rathod, Advani, & Gonsai, 2018).

2.1.1. Advanced Encryption Standard Algorithm: The Advanced Encryption Standard (AES) is a symmetric encryption algorithm. AES is applied to the data, which has fixed-size blocks of 128 bits for the encryption and decryption process and supports key sizes of 128, 192, and 256 bits; based on this key, the number of rounds in encryption and decryption is followed: Ten rounds of repetition for 128-bit keys. Twelve rounds of repetition for 192-bit keys. Fourteen rounds of repetition for 256-bit keys. The encryption process involves multiple rounds of transformations that enhance the security and complexity of the ciphertext. Each round of the encryption process requires the following four types of operations: Sub-Bytes, ShiftRows, MixColumns, and XOR Round key. Decryption is the reverse process of encryption, and using inverse functions, only the (n-1) round process has only three operations except MixColumns. For a better understanding, the algorithm is explained below with the help of a block diagram, as shown in Figure 3.

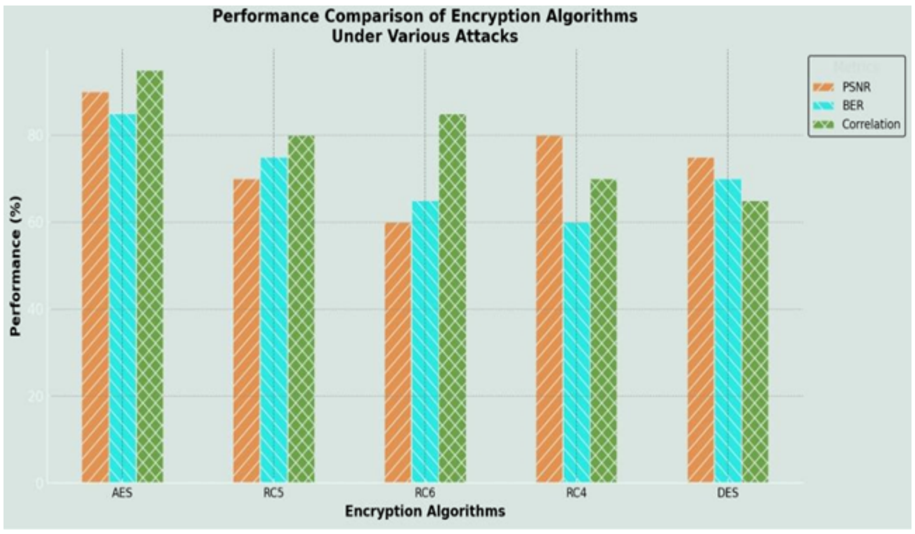


Figure 3: Graph showing performance comparison between stream & block cipher.

AES is a robust and efficient encryption standard widely adopted for securing sensitive data. Its flexibility in key size and structured approach to data transformation ensures strong protection against various cryptographic attacks (Hasija, Rustagi, Rathore, & Gupta, 2024).

2.1.2. Data Encryption Standard Algorithm: The Data Encryption Standard (DES) is a symmetric-key block cipher selected as a federal standard in the United States in 1977. It is applied to plain text, which has 64 bits for encryption and cipher text (64 bits) for decryption. The key that has a 64-bit size was selected. The encryption and decryption process has 16 rounds each. Each block of 64 bits is divided into 32 bits each. If encryption is done individually for each 64-bit block, then the encryption mode is called Electronic Code Block (ECB). The Algorithm of DES is given below, and algorithm 2 represents the DES Encryption Process flow chart (Kaur & Sodhi, 2016).

2.1.3. RC5: RC5 is a symmetric key block cipher encryption algorithm designed by Ron Rivest, notable for its simplicity due to its reliance on primitive computer operations such as addition, subtraction, bitwise XOR, and circular shift. This algorithm operates on variable block sizes, typically 16, 32, or 64 bits, and supports variable key sizes ranging from 0 to 2040 bits. The algorithm is executed in rounds, with the number of rounds determined by the key size. For a better understanding, the algorithm is explained below with the help of a block diagram, as shown in algorithm 3.

RC5 is designed to provide a high level of security against various attacks, including brute-force attacks and differential cryptanalysis. However, it is vulnerable to side-channel attacks, such as timing attacks and power analysis attacks, which exploit information leaked through the implementation of the algorithm rather than targeting the algorithm itself (Rivest, 1994).

Algorithm 1 AES

```
ENCRYPTION
Read video frame-by-frame and convert each frame to a byte array.
Define the AES key (128-bit, 16-byte) and initialize.
Perform key expansion to generate the key schedule for all rounds.
for each frame do
    Convert frame to a 1D array of bytes (plaintext).
    Divide plaintext into 16-byte blocks.
    for each block do
        Add the initial round key (first 16 bytes of expanded key).
        for round = 1 to 9 do
            SubBytes: Substitute each byte using the AES S-Box.
            ShiftRows: Shift rows of the state matrix cyclically.
            MixColumns: Mix columns using matrix multiplication.
            AddRoundKey: XOR with the expanded key.
        end for
        Final Round (Round 10) without MixColumns:
        SubBytes: Substitute each byte using the AES S-Box.
        ShiftRows: Shift rows of the state matrix cyclically.
        AddRoundKey: XOR with the last round key.
        Store the encrypted block.
    end for
    Reassemble the encrypted blocks into an encrypted frame.
end for
Write the encrypted frame back to the video file or save as a new file.
DECRYPTION
Read the encrypted video frame-by-frame and convert each frame to a byte array.
Define the AES key (128-bit, 16-byte) and initialize.
Perform key expansion to generate the key schedule for all rounds.
for each encrypted frame do
    Convert frame to a 1D array of bytes (ciphertext).
    Divide ciphertext into 16-byte blocks.
    for each block do
        Add the last round key (last 16 bytes of expanded key).
        InvShiftRows: Reverse the row shifts of the matrix.
        InvSubBytes: Reverse substitution using the inverse AES S-Box.
        AddRoundKey: XOR with the key for round 9.
        for round = 9 down to 1 do
            AddRoundKey: XOR with the expanded key.
            InvMixColumns: Reverse mixing of columns.
            InvShiftRows: Reverse the row shifts of the matrix.
            InvSubBytes: Reverse substitution using the inverse AES S-Box.
        end for
        Add the initial round key (first 16 bytes of expanded key).
        Store the decrypted block.
    end for
    Reassemble the decrypted blocks into a decrypted frame.
end for
Write the decrypted frame back to the video file or save as a new file.
```

Algorithm 2 DES

```
ENCRYPTION
INPUT: VIDEO.mp4 = DATA_IN, INITIAL_KEY = '133457799BBCDFF1', PERM_COMB1, INIT_PERM, NUMOFSHIFT, PERM_COMB2, BITSEL_ARR, PERM_COMB3, INV_PERM
OUTPUT: CIPHER TEXT = DATA_OUT
STEP 1: EXTRACT FRAMES
Read VIDEO and convert each frame to plain text DATA_IN.
STEP 2: KEY TRANSFORMATION
Transform the 64-bit key to 56-bit by discarding every 8th bit.
Split into two 28-bit halves, apply circular shifts as per NUMOFSHIFT.
Generate 16 subkeys using PERM_COMB2.
STEP 3: INITIAL PERMUTATION (IP)
Convert DATA_IN to binary and apply INIT_PERM.
STEP 4: SPLIT INTO HALVES
Divide permuted input into LPT and RPT.
STEP 5: ENCRYPTION ROUNDS
for round = 1 to 16 do
    Apply subkey from order K1 to K16.
    Process LPT and RPT through DES rounds.
end for
STEP 6: FINAL PERMUTATION (FP)
Combine LPT and RPT, then apply INV_PERM.
STEP 7: CONVERT TO CIPHER TEXT
Convert binary data to hex and output DATA_OUT.
DECRYPTION
INPUT: CIPHER TEXT = DATA_OUT, INIT_PERM, BITSEL_ARR
OUTPUT: RETRIEVED VIDEO.mp4
STEP 1: KEY TRANSFORMATION
Transform the 64-bit key to 56-bit by discarding every 8th bit.
Split into two 28-bit halves and apply circular shifts as per NUMOFSHIFT.
STEP 2: INITIAL PERMUTATION (IP)
Convert DATA_OUT to binary and apply INIT_PERM.
STEP 3: SPLIT INTO HALVES
Divide permuted input into LCT and RCT.
STEP 4: DECRYPTION ROUNDS
for round = 1 to 16 do
    Apply key from the end (K16 for round 1, K1 for round 16).
    Process RCT and LCT through the same steps as encryption but in reverse order.
end for
STEP 5: FINAL PERMUTATION (FP)
Combine LCT and RCT, then apply INV_PERM.
STEP 6: CONVERT TO FRAMES
Convert binary data back to frames and reassemble into VIDEO.mp4.
```

Algorithm 3 RC5

```
ENCRYPTION
INPUT: VIDEO.MP4, KEY = ’2B7E151628AED2A6ABF7158809CF4F3C’
OUTPUT: CIPHER_TEXT = DATA_OUT
STEP 1: EXTRACT FRAMES
VIDEO = VIDEOREADER(’VIDEO.MP4’)
for each frame do
    DATA_IN = FRAME(:)
end for
STEP 2: KEY EXPANSION
INITIALIZE L, S, P BASED ON KEY
EXPAND L TO FIT S AND P
STEP 3: ENCRYPTION ROUNDS
for each round do
    UPDATE A, B WITH S AND P
end for
STEP 4: FINAL MIXING
A = A + S(1)
B = B + S(2)
STEP 5: CONVERT TO ENCRYPTED FRAMES
CIPHER_TEXT = DATA_OUT
ENCRYPTED_FRAMES = CONVERT_TO_FRAMES(CIPHER_TEXT)
FUNCTION CONVERT_TO_FRAMES(CIPHER_TEXT)
        FRAME_SIZE = 30
        NUM_FRAMES = CEIL(LENGTH(CIPHER_TEXT) / FRAME_SIZE)
for each frame do
    ENCRYPTED_FRAMESI = CIPHER_TEXT(START_IDX:END_IDX)
end for
PRODUCE FINAL CIPHER TEXT DATA_OUT
DECRYPTION
INPUT: CIPHER_TEXT = DATA_OUT, KEY EXPANDED, S1, E-TABLE, L-TABLE, MUL-MAT
OUTPUT: VIDEO.MP4
KEY EXPANSION
INITIALIZE L, S, P BASED ON KEY
EXPAND L TO FIT S AND P
DECRYPTION ROUNDS
for each round (reverse order) do
    UPDATE B, A WITH S AND P
end for
FINAL MIXING
B = B - S(2)
A = A - S(1)
STEP 6: CONVERT TO PLAINTEXT
PLAINTEXT = ...
STEP 7: CONVERT TO FRAMES AND VIDEO
DECRYPTED_FRAMES = CONVERT_TO_FRAMES(PLAINTEXT)
VIDEO.MP4 = RECONVERT_FROM_FRAMES(DECRYPTED_FRAMES)
```

2.1.4. RC6: RC6 is a symmetric-key block cipher encryption algorithm that offers a unique combination of simplicity, speed, and security. Building upon the RC5 encryption algorithm, RC6 incorporates distinct differences that enhance its security features. As a variant of the widely used and secure Advanced Encryption Standard (AES) algorithm, RC6 is precisely defined as RC6-w/r/b, where w represents the word size in bits, r denotes the number of encryption rounds (a non-negative integer), and b signifies the length of the encryption key in bytes. Notably, RC6 employs a fixed block size of 128 bits, differing from the variable block size used in RC5. Furthermore, RC6 accommodates 128, 192, and 256-bit variable key sizes. While RC6 demonstrates robust security features, it remains vulnerable to brute-force attacks. One area for improvement of the RC6 algorithm is the disparity between its encryption and decryption processes, which may introduce complexity in its implementation. For a better understanding, the algorithm is explained below with the help of a block diagram, as shown in algorithm 4 (Mishra, Gupta, Krishna Murthy, & Pal, 2021).

Algorithm 4 RC6

```
1: ENCRYPTION
2: INPUT: VIDEO.MP4, KEY = '2B7E151628AED2A6ABF7158809CF4F3C'
3: OUTPUT: CIPHER_TEXT = DATA_OUT
4: STEP 1: Extract Frames from the Video
5: VIDEO = VIDEOREADER('input_video.mp4')
6: FRAME_COUNT = VIDEO.NUMFRAMES
7: for each frame do
8:     DATA_IN = FRAME(:)
9: end for
10: STEP 2: Key Expansion using RC6 Algorithm
11: S = RC6_KEY_EXPANSION(KEY)
12: STEP 3: Encrypt Data
13: ENCRYPTED_DATA = RC6_ENCRYPT(DATA_IN, S)
14: STEP 4: Convert Cipher Text to Encrypted Frames
15: ENCRYPTED_FRAMES = CONVERT_TO_FRAMES(CIPHER_TEXT)
16: STEP 5: Result
17: Encrypted video frames DATA_OUT.
18: DECRYPTION
19: INPUT: CIPHER_TEXT = DATA_OUT, KEY
20: OUTPUT: RETRIEVED VIDEO.MP4
21: STEP 1: Key Expansion (same as encryption)
22: S = RC6_KEY_EXPANSION(KEY)
23: STEP 2: Decrypt Data
24: DECRYPTED_DATA = RC6_DECRYPT(ENCRYPTED_DATA, S)
25: STEP 3: Result
26: Decrypted plain text RET_DATA.
27: STEP 4: Convert Decrypted Text to Frames and Rebuild Video
28: Convert DECRYPTED_DATA back to frames and reassemble the video.
```

2.2. Stream Cipher

This data encryption and decryption method is done on a stream of data. It has two main components: a keystream generator and a mixing function. The keystream generator is the primary unit in the stream cipher encryption technique, while the mixing function is usually just an XOR function. For example, if the keystream generator produces a series of zeroes, the output ciphered stream will be identical to the original plain text (Rathod et al., 2018).

2.2.1. RC4: RC4 was designed by Ron Rivest for RSA Security in 1987; a stream cipher-type algorithm that processes a unit or input data byte at a time. This algorithm is based on random permutation. RC4 algorithm is simple and relatively easy to implement. In this, we take a key of length (40 to 2040 bits) as input to a pseudo-random bit generator that produces a random key stream of bits, which are XOR with a data stream byte at a time to encrypt it and XOR again with the same key stream to decrypt it. For a better understanding, the algorithm is explained below with the help of a block diagram, as shown in algorithm 5 (Liu, Jin, & Li, 2024).

It has two parts-

- KSA - It generates a key stream with the help of permutation of S Box.
- PRGA- Determines the value of K and Performs the XOR operation to generate cipher text.

A weak point of this algorithm is a related key vulnerability, which applies when part of the key presented to the KSA is exposed to the attacker.

3. Simulation Setup Parameters

3.1. Setup Parameters

Simulation Setup Parameters can refer to the configuration settings and parameters used to simulate video data processing. These parameters are critical for defining how the video processing simulation is conducted and can vary depending on the specific goals and requirements of the simulation (Mithlesh, Shukla, & Sharma, 2016). Here are some simulation setup parameters for video processing:

Table 1: Setup Parameters.

Processor	Intel(R) core i5 -8350U CPU @1.70GHz	
RAM	16 GB	
SSD	512 GB	
Language	Java 1.8.0_351	
Text editor	Eclipse IDE, Version: 2023-03 (4.27.0)	
Frame rate	30	
Video	Video 1(.mp4)	09 sec - 280 frames - 1280*720
	Video 2(.mp4)	20 sec - 626 frames - 360*240
	Video 3(.mp4)	13 sec - 328 frames - 352*192

Algorithm 5 RC4

```
ENCRYPTION
INPUT: VIDEO.mp4, KEY = '250'
OUTPUT: CIPHER_TEXT = DATA_OUT
STEP 1: Extract Frames
VIDEO = VIDEOREADER('input_video.mp4')
FRAME_COUNT = VIDEO.NUMFRAMES
for each frame $i = 1$ to FRAME_COUNT do
    FRAME = READ(VIDEO, $i$)
    DATA_IN = FRAME(:)
end for
STEP 2: Initialize S and Perform KSA (Key Scheduling Algorithm)
Initialize $S = [0, 1, 2, \ldots, 255]$
Set seed = 250
KEY = RANDI([0, 1], 256)                    ▷ Random key generation of 256 bytes
for $i = 0$ to 255 do
    $J = \text{MOD}(J + S(i+1) + KEY(\text{MOD}(i, \text{LENGTH}(KEY)) + 1), 256)$
    Swap $S(i+1)$ and $S(J+1)$
end for
STEP 3: PRGA and XOR with DATA_IN to get DATA_OUT
Initialize $I = 0, J = 0$
for $i = 1$ to NUMEL(DATA_IN) do
    $I = \text{MOD}(I + 1, 256)$
    $J = \text{MOD}(J + S(I+1), 256)$
    Swap $S(I+1)$ and $S(J+1)$
    CT(i) = BITXOR(DATA_IN(i), S(MOD(S(I+1) + S(J+1), 256) + 1))
end for
STEP 4: Convert CIPHER TEXT to Encrypted Frames
ENCRYPTED_FRAMES = CONVERT_TO_FRAMES(DATA_OUT)
DECRYPTION
INPUT: CIPHER_TEXT = DATA_OUT, KEY = '250'
OUTPUT: RETRIEVED VIDEO.mp4
STEP 1: Initialize S and Perform KSA (Same as Encryption)
Initialize $S = [0, 1, 2, \ldots, 255]$
Set seed = 250
KEY = RANDI([0, 1], 256)                    ▷ Random key generation of 256 bytes
for $i = 0$ to 255 do
    $J = \text{MOD}(J + S(i+1) + KEY(\text{MOD}(i, \text{LENGTH}(KEY)) + 1), 256)$
    Swap $S(i+1)$ and $S(J+1)$
end for
STEP 2: PRGA and XOR with CIPHER_TEXT to retrieve DATA_IN
Initialize $I = 0, J = 0$
for $i = 1$ to NUMEL(DATA_OUT) do
    $I = \text{MOD}(I + 1, 256)$
    $J = \text{MOD}(J + S(I+1), 256)$
    Swap $S(I+1)$ and $S(J+1)$
    PTT(i) = BITXOR(DATA_OUT(i), S(MOD(S(I+1) + S(J+1), 256) + 1))
end for
STEP 3: Convert PTT to Frames and Rebuild Video
Convert PTT back to frames and reassemble into RETRIEVED VIDEO.mp4.
```

3.2. Performance Parameters

The encryption mechanisms are evaluated using various parameters such as Peak Signal-to-Noise Ratio (PSNR), Mean Square Error (MSE), Correlation Coefficient (CC), Structural Similarity Index Measure (SSIM), Features Similarity Index Measure (FSIM), Snapshots and Histograms.

3.2.1. Mean Squared Error (MSE): MSE measures the average squared difference between the original and the encrypted/decrypted image pixels. It is a basic metric for quantifying the error or distortion introduced by the encryption and decryption process. Lower MSE values indicate less error and higher similarity. In the context of encryption, MSE helps quantify the distortion between the original and decrypted images.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \tag{1}$$

Where:

- $\sum_{i=1}^{N}$: The summation symbol indicates that the expression should be summed over all pixels from $i = 1$ to $i = N$. This means that the error is calculated for each corresponding pixel pair and then summed.
- x_i: The pixel value at position *i* in the reference image (original image).
- y_i: The pixel value at position iii in the test image (processed or reconstructed image).

3.2.2. Peak Signal-to-Noise Ratio (PSNR): PSNR is widely used to measure the quality of an image by comparing the original and the encrypted (or decrypted) images. It is beneficial in scenarios where the goal is to maintain high fidelity to the original image, such as in image compression and encryption. Higher PSNR values indicate better image quality. In the context of encryption, PSNR is used to evaluate the visual similarity between the original and decrypted images.

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{(\text{MAX})^2}{\text{MSE}}\right) \tag{2}$$

MSE is the average squared difference between the pixel values of the original image and the reconstructed image. It quantifies the average error per pixel between the two images. A lower MSE indicates that the reconstructed image is closer to the original. MAX represents the maximum possible pixel value, which normalizes the MSE to ensure the PSNR is independent of the image scale.

3.2.3. Correlation Coefficient: The correlation coefficient measures the relationship between the pixel values of the original and encrypted images. Ideally, a secure encryption scheme should result in a low correlation between the encrypted and original images, indicating that the encrypted image is significantly different from the original. Values close to 0 indicate low correlation, suggesting effective encryption. Values close to ±1 indicate a high correlation, implying poor encryption quality.

$$\text{Correlation} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \tag{3}$$

Where:

- x_i: The pixel value at position i in the reference image (original image).
- y_i: The pixel value at position i in the test image (processed or reconstructed image).
- $\bar{x}$: The mean (average) of all pixel values in the reference image.
- $\bar{y}$: The mean (average) of all pixel values in the test image.

3.2.4. Structural Similarity Index (SSIM): SSIM is used to measure the similarity between two images by comparing their structural information. It considers luminance, contrast, and structure. SSIM values range from 0 to 1, with values closer to 1 indicating high similarity. In encryption, SSIM can be used to evaluate how well the decrypted image retains the structural features of the original image.

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{4}$$

Where:

- x & y : The reference image and the test image, respectively. These are the two images being compared.
- μ_x: The mean (average) pixel value of the reference image. It represents the average luminance of the reference image.
- μ_y: The mean (average) pixel value of the test image. It represents the average luminance of the test image.
- σ_x^2: The variance of the pixel values in the reference image. It measures the spread or dispersion of the pixel values around the mean.
- σ_y^2: The variance of the pixel values in the test image. It measures the spread or dispersion of the pixel values around the mean.
- σ_{xy}: The covariance of the pixel values between the reference image and the test image. It measures how much the pixel values in the two images vary together. C1 and C2: A small constant to avoid division by zero.

3.2.5. Feature Similarity Index (FSIM): FSIM evaluates the similarity between images based on low-level features such as edges and textures. Higher FSIM values indicate that the images are perceptually similar. In encryption, FSIM can be used to assess how closely the decrypted image resembles the original.

$$\text{FSIM}(x, y) = \frac{\sum_i \text{SL}(i) \cdot \text{PCm}(i)}{\sum_i \text{PCm}(i)} \tag{5}$$

Where:

- SL(i) Local similarity measure at the i-th pixel.
- PCm(i) The phase congruency magnitude at the i-th pixel.

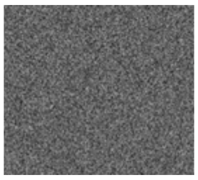
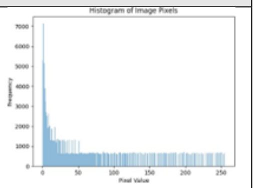
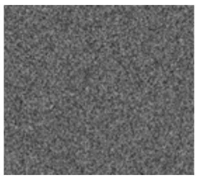
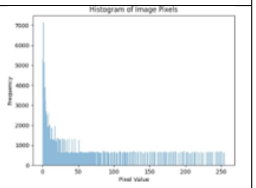
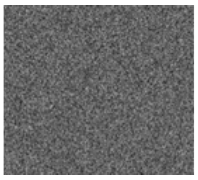
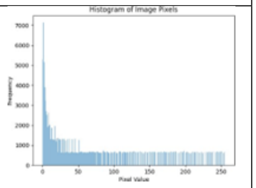
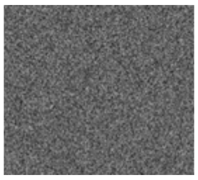
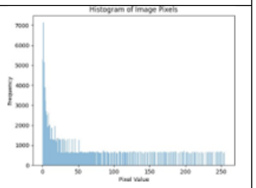
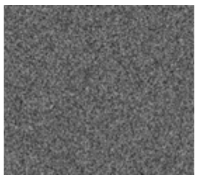
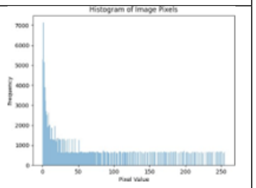
4. Results

The performance of the algorithms is accessed in the presence and absence of various attacks, such as noise attacks and geometric attacks. The results are demonstrated based on different analyses, such as quantitative, qualitative and robustness.

4.1. Qualitative Analysis

Tables 2, 3 and 4 represent the original frames and encrypted frames with the histogram of encrypted videos for corresponding algorithms (Sethi & Vijay, 2013).

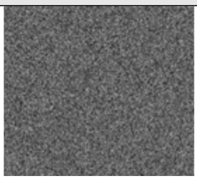
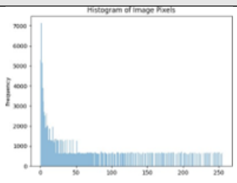
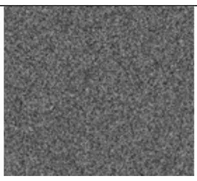
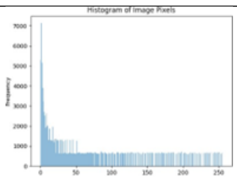
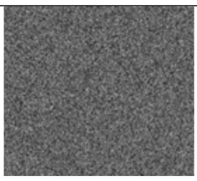
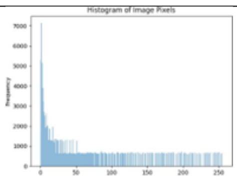
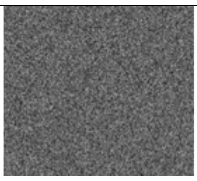
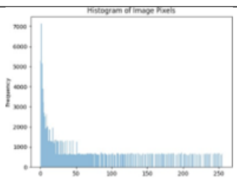
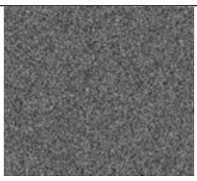
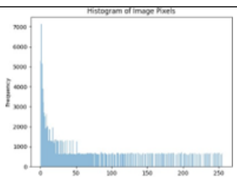
Table 2: Video 1 (9 sec).

	Original image	**Encrypted Image**	**Histogram**
AES			
DES			
RC4			
RC5			
RC6			

4.2. Quantitative analysis

When analysing the security of image encryption schemes, several metrics are commonly used to assess the quality and effectiveness of the encryption. These metrics help determine how well the encrypted image resists attacks and maintains its integrity. Key metrics include PSNR, Correlation, FSIM, SSIM and MSE (Sethi & Vijay, 2013). Table 5 shows values calculated for different performance parameters between Original and Encrypted Frames-

Table 3: Video 2 (20 sec).

	Original image	**Encrypted Image**	**Histogram**
AES			
DES			
RC4			
RC5			
RC6			

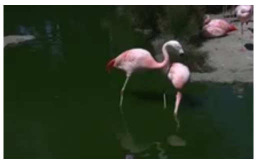
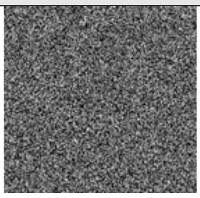
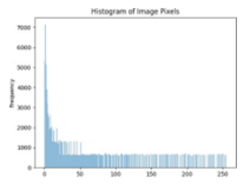
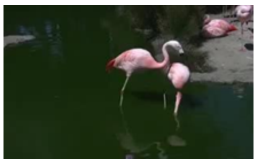
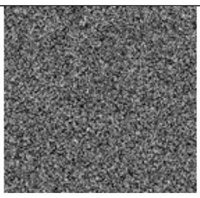
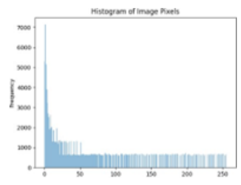
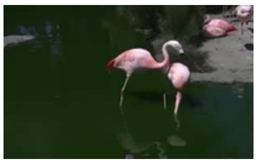
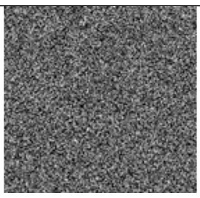
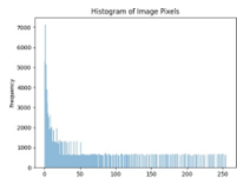
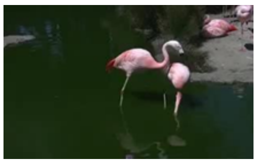
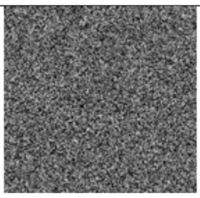
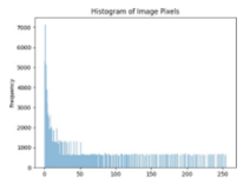
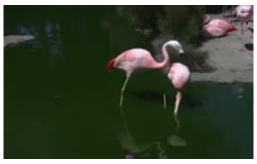
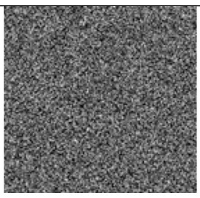
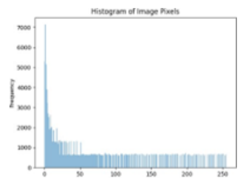
As observed from the table, the RC5 algorithm offers the best result in terms of PSNR. For FSIM, the AES algorithm provides the best performance, and the RC5 algorithm yields the highest result regarding SSIM and Correlation coefficient.

4.3. Robustness analysis

Robustness analysis of an encryption algorithm is a crucial process that evaluates its ability to withstand various attacks and maintain its security properties. It involves testing the algorithm against different types of attacks, such as noise attacks, flip attacks, and cryptanalysis, to identify vulnerabilities and weaknesses. The importance of robustness analysis lies in ensuring sensitive data's confidentiality, integrity, and authenticity. A robust encryption algorithm can protect against unauthorized access, tampering, and eavesdropping, thereby maintaining the trust and confidence of users. In today's digital landscape, where cyberattacks are increasingly common, robust encryption algorithms are essential for safeguarding sensitive information and preventing financial losses, reputational damage, and legal liabilities. In this paper, we have analyzed the robustness of block and stream ciphers by applying various attacks on encrypted video frames, as detailed in the following sections (Pal & Verma, 2016).

4.3.1. Noise attack:

Table 4: Video 3 (13 sec).

	Original image	Encrypted Image	Histogram
AES			
DES			
RC4			
RC5			
RC6			

- Salt and Pepper: In this analysis, random noise is introduced to the encrypted video frames by adding random noise to them, Specifically, researchers have added 10% salt and pepper noise effectively replacing the 10%-pixel value of encrypted frames with the either maximum value (white) or with the minimum possible value (Black).

  This type of noise is also called impulse noise. This attack simulates the effects of noisy transmission channels or intentional data tampering, allowing us to evaluate the robustness of the encryption algorithms against such disruptions.

  The results of this attack can be seen in 6, which illustrates the impact of this attack by calculating the Peak Signal-to-Noise Ratio (PSNR), Bit Error Rate (BER), and Correlation Coefficient values between the Original and decrypted frames, which reveal a significant degradation in video quality and integrity (Malik, Gupta, & Dhall, 2020).

  As observed from the table after introducing Salt & Pepper noise, the RC4 algorithm offers the best result in terms of PSNR. For BER, the RC5 algorithm provides the best performance, and the RC6 algorithm yields the highest result regarding the Correlation coefficient.

Table 5: Values for Different Performance Metrics.

	Videos	DES	AES	RC4	RC5	RC6
PSNR	Vid1	9.46395	9.46362	9.46505	9.46369	9.46451
	Vid2	8.06487	8.06481	8.06565	8.06465	8.06575
	Vid3	6.95859	6.95852	6.9562	6.9565	6.95691
Average		8.16247	8.16232	8.1623	8.16161	8.16239
MSE	Vid1	7417.8619	7418.1657	7415.557	7418.1301	7416.6881
	Vid2	6246.1206	6246.3134	6243.5166	6244.9952	6242.4078
	Vid3	10519.996	10519.79	10525.228	10523.206	10521.554
Average		8061.3263	8061.4233	8061.4341	8062.1106	8060.2168
FSIM	Vid1	0.025492	0.02556	0.02572	0.0256	0.025739
	Vid2	0.012048	0.01189	0.012327	0.01214	0.01251
	Vid3	0.01196	0.01155	0.01103	0.01127	0.01153
Average		0.0165	0.01633	0.01636	0.01634	0.01659
SSIM	Vid1	0.01273	0.01277	0.01286	0.01279	0.012877
	Vid2	0.012048	0.011897	0.01232	0.01214	0.0125
	Vid3	0.011961	0.01155	0.011035	0.01127	0.01153
Average		0.01224	0.012073	0.012076	0.012	0.01222
Correlation	Vid1	0.000378	0.00046	0.000614	0.000507	0.00059
	Vid2	0.002406	0.002234	0.002723	0.002513	0.00294
	Vid3	0.00398	0.00338	0.002712	0.00296	0.00339
Average		0.00225	0.00206	0.00201	0.00199	0.00231

Table 6: Values for different performance metrics after noise attack.

	Videos	DES	AES	RC4	RC5	RC6
PSNR	vid1	8.3114	8.31382	8.33358	8.44333	8.3158
	vid2	7.98287	8.0171	8.065505	5.94676	8.01946
	vid3	6.95917	10.9633	7.01309	11.26985	7.00464
Average		7.75115	9.09808	9.1208	8.55331	7.77996
BER	vid1	0.49952	0.49961	0.49992	0.46328	0.49966
	vid2	0.49984	0.49983	0.5	0.4941	0.49981
	vid3	0.50026	0.42185	0.4974	0.417	0.49784
Average		0.49988	0.47376	0.49911	0.45815	0.4991
Correlation	vid1	-0.00158	-0.00052	0.00153	-0.0064	-0.00094
	vid2	0.0068	0.01063	0.00276	0.0132	0.01078
	vid3	0.0036	0.0001	0.009647	-0.00758	0.0087
Average		0.0029	0.0034	0.00146	-0.00028	0.00618

- Gaussian Attack: In this attack like salt and pepper 10% Gaussian noise is added to encrypted video frames effectively simulating the effects of noisy transmission channels or intentional data tampering. This attack involves the generation of noise by randomly sampling from a Gaussian distribution, and the resulting noise values are added to the 10%-pixel values of the encrypted video frames. The attack is characterized by the mean, variance and standard deviation of Gaussian distribution.

The results of this attack can be seen in Table 7 below, which illustrates the impact of this attack as like above by calculation of PSNR, BER and Correlation coefficient between Original and decrypted frames.

Table 7: Values for different performance metrics after gaussian attack.

	Videos	DES	AES	RC4	RC5	RC6
PSNR	vid1	8.33288	8.31321	8.3061	8.4431	8.33539
	vid2	8.06506	8.01596	7.91709	5.94694	8.06501
	vid3	6.96094	11.24389	7.0137	11.27014	6.95867
Average		7.78629	9.19102	7.74563	8.55339	7.78635
BER	vid1	0.49996	0.49962	0.49915	0.46328	0.4999
	vid2	0.50001	0.49983	0.49982	0.4941	0.49996
	vid3	0.50026	0.41737	0.49745	0.41708	0.50032
Average		0.50008	0.47228	0.49881	0.45815	0.50006
Correlation	vid1	0.00198	-0.00073	-0.00393	-0.00679	0.00189
	vid2	0.00236	0.01062	-0.00133	0.01414	0.00242
	vid3	0.00365	-0.00101	0.00953	-0.00753	0.00357
Average		0.00266	0.00295	0.00142	-0.00061	0.00263

As observed in the table after introducing Gaussian Noise, the AES algorithm offers the best PSNR and correlation coefficient results. For BER, the RC5 algorithm provides the best performance.

- Flip Attack: In a flip attack, specific bits in the encrypted video frames are intentionally altered, simulating the effects of bit-flipping on the encrypted data. When these modified encrypted frames are decrypted, the bit alterations can cause errors in the decryption process, leading to a distorted or corrupted decrypted video. This attack is particularly relevant in scenarios where encrypted video is transmitted over networks susceptible to bit-flipping errors or stored on devices vulnerable to malicious tampering.

  The results of this attack can be seen in Table 8 below, which illustrates the impact of this attack by calculating the PSNR, BER and Correlation Coefficient between the original and decrypted frames.

  As observed in the table 8, after introducing a flip attack, the AES algorithm offers the best PSNR. For BER, the RC5 algorithm provides the best performance. The RC6 algorithm demonstrates the best results regarding the correlation coefficient.

- Rotation Attack: In this attack, the encrypted video frames are rotated by 90°, simulating the effects of intentional data tampering or transmission errors. This attack involves rotating the encrypted frames by a fixed angle, resulting in a distorted decrypted video.

  The results of this attack can be seen in table 9 below, which illustrates the impact of this attack, like above, by calculating the PSNR, BER, and Correlation Coefficient between the Original and decrypted frames (Zheng & Zhang, 2020).

Table 8: Values for different performance metrics after flip attack.

	Videos	DES	AES	RC4	RC5	RC6
PSNR	vid1	8.31364	8.31366	8.30395	8.44301	8.31318
	vid2	7.98345	8.01687	7.90778	5.94687	8.01783
	vid3	6.95991	11.23835	7.01343	11.26976	7.00469
Average		7.75234	9.18963	7.74172	8.55321	7.77857
BER	vid1	0.49947	0.49965	0.49913	0.46328	0.49964
	vid2	0.49982	0.49984	0.49989	0.4941	0.49982
	vid3	0.50028	0.41742	0.4974	0.41708	0.49783
Average		0.49986	0.4723	0.49881	0.45816	0.4991
Correlation	vid1	-0.00105	–0.00052	-0.00433	-0.00696	-0.00123
	vid2	0.00696	0.0106	-0.00175	0.0134	0.010641
	vid3	0.00334	-0.00073	0.00938	-0.00777	0.008803
Average		0.00308	0.00328	0.00109	-0.00044	0.00606

Table 9: Values for different performance metrics after flip attack.

	Videos	DES	AES	RC4	RC5	RC6
PSNR	vid1	8.34673	8.44105	8.33396	8.44507	8.44171
	vid2	5.95156	5.95189	8.06509	5.95045	5.95142
	vid3	11.26794	11.25221	6.9576	11.2669	11.2612
Average		8.52208	8.54839	7.78555	8.55417	8.55145
BER	vid1	0.4634	0.46336	0.49992	0.4633	0.46334
	vid2	0.49412	0.49409	0.50024	0.49411	0.49411
	vid3	0.41717	0.41729	0.50032	0.4172	0.41724
Average		0.45823	0.45825	0.50009	0.4582	0.45823
Correlation	vid1	-0.0012	-0.00183	0.00185	0.001024	0.00014
	vid2	0.00124	0.01679	0.00267	0.00727	-0.0003
	vid3	0.00048	0.00798	0.00332	0.0005	0.00058
Average		0.00017	0.00769	0.00261	0.00075	0.00014

As observed in the table, after introducing a Rotation Attack, the RC5 algorithm offers the best PSNR. For BER, the RC5 algorithm provides the best performance. The AES algorithm demonstrates the best results regarding the correlation coefficient.

- JPEG Attack: In this attack, the encrypted video frames are compressed using JPEG compression, simulating the effects of lossy compression on the encrypted data. When applied to encrypted data, JPEG compression can cause errors in the decryption process, leading to a distorted or corrupted decrypted video. This attack is particularly relevant in scenarios where encrypted video is transmitted over networks or stored on devices with limited storage capacity.

  The results of this attack can be seen in Table 10, which illustrates the impact of this attack, like above, by calculating the PSNR, BER, and Correlation Coefficient between the Original and decrypted frames (Aydemir, Temizel, & Temizel, 2018).

Table 10: Values for different performance metrics after JPEG attack.

	Videos	DES	AES	RC4	RC5	RC6
PSNR	vid1	8.43978	8.63858	8.33176	8.44413	8.44031
	vid2	5.95176	6.16147	8.00836	5.9514	5.9535
	vid3	11.2412	11.0026	7.01706	11.2418	11.2456
Average		8.54426	8.60089	7.78573	8.54578	8.5465
BER	vid1	0.46343	0.46076	0.49922	0.46338	0.46341
	vid2	0.49413	0.48959	0.4995	0.4942	0.49413
	vid3	0.41743	0.4209	0.49815	0.41746	0.41736
Average		0.45833	0.45708	0.49896	0.45835	0.4583
	vid1	0.0009	-0.0011	-0.00129	0.00169	-0.00038
Correlation	vid2	0.00047	0.00244	0.00274	0.00031	0.001081
	vid3	-0.00036	0.00159	0.00698	0.00022	-0.00163
Average		0.00034	0.00097	0.00199	0.00074	-0.00031

As observed in the table, after introducing a JPEG Attack, the AES algorithm offers the best PSNR and BER. The RC4 algorithm demonstrates the best results regarding the correlation coefficient.

5. Conclusion

Based on the analysis of various encryption algorithms under different attacks, the following conclusions can be drawn:

- AES: Demonstrates superior performance in PSNR and BER under Gaussian Noise and JPEG attacks and in Correlation under Rotation and Gaussian Noise attacks.
- RC5: Excels in BER under Salt & Pepper, Gaussian Noise, Flip, and Rotation attacks; also performs best in PSNR under Rotation attacks.
- RC6: Performs best in Correlation under Salt & Pepper and Flip attacks.
- RC4: Offers the best PSNR under Salt & Pepper attacks and best Correlation under JPEG attacks.

Overall, AES and RC5 are the most robust algorithms across different metrics and attacks, with RC6 and RC4 showing specific strengths in certain conditions. The study concludes that block ciphers offer superior security & robustness.

References

A., A. K., & Manikandan, L. C. (2020). A study on cryptographic techniques. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, *6*(4), 321–327. doi: 10.32628/CSEIT206453

Aydemir, A. E., Temizel, A., & Temizel, T. T. (2018). The Effects of JPEG and JPEG2000 Compression on Attacks Using Adversarial Examples. *arXiv preprint arXiv:1803.10418*. doi: https://arxiv.org/abs/1803.10418

Baeldung. (2024). *Working with images in Java.* (Retrieved June 23, 2024, from https://www.baeldung.com/java-images)

Hasija, U., Rustagi, P., Rathore, N., & Gupta, V. (2024). Cryptographic foundations: A comprehensive review of block cipher and stream cipher concepts. In *Proceedings of the 2024 11th international conference on computing for sustainable global development (indiacom).* doi: 10.23919/INDIACom61295.2024.10498260

Kaur, N., & Sodhi, S. (2016). Data Encryption Standard Algorithm (DES) for Secure Data Transmission. In *Proceedings of the international conference on advances in emerging technology (icaet)* (pp. 31–37). doi: https://ijcaonline.org/proceedings/icaet2016/number2/25887-2314

Kaushik, B., Malik, V., & Saroha, V. (2023). A review paper on data encryption and decryption. *International Journal for Research in Applied Science & Engineering Technology*, *11*(4), 1986–1992. doi: 10.22214/ijraset.2023.50101

Liu, S., Jin, Z., & Li, Y. (2024). Research on efficient stream cipher design in big data environment. In *Icmlc '24: Proceedings of the 2024 16th international conference on machine learning and computing.* doi: 10.1145/3651671.3651754

Malik, A., Gupta, S., & Dhall, S. (2020). Analysis of traditional and modern image encryption algorithms under realistic ambiance. *Multimedia Tools and Applications*, *79*(37-38), 27779–27810. doi: 10.1007/s11042-020-09279-6

Mishra, G., Gupta, I., Krishna Murthy, S. V. S. S. N. V. G., & Pal, S. K. (2021). Deep learning based cryptanalysis of stream ciphers. *Defence Science Journal*, *71*(4), 499–506. doi: 10.14429/dsj.71.16209

Mithlesh, C. S., Shukla, D., & Sharma, M. (2016). Video to image conversion techniques video frame extraction. *International Journal of Emerging Science and Engineering (IJESE)*, *4*(3), 1–2. doi: https://www.ijese.org/wp-content/uploads/Papers/v4i3/C1064024316.pdf

Pal, G., & Verma, V. (2016). Image encryption techniques under various noise attacks: A survey. *International Journal of Software & Hardware Research in Engineering*, *4*(12), 48–56. doi: https://ijournals.in/wp-content/uploads/2017/07/7.41113-Garima.compressed.pdf

Petrosyan, A. (2024). *Number of Data Compromises and Impacted Individuals in U.S. 2005–2023.* (Retrieved June 23, 2024, from https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/)

Rathod, C., Advani, N., & Gonsai, A. (2018). Comparative analysis of encryption and decryption algorithms for audio. *International Journal of Recent Research Aspects*, *5*(2), 32–35. doi: https://www.ijrra.net/Vol5issue2.php

Repository, M. C. (2024). *Maven central repository.* (Retrieved June 20, 2024, from https://mvnrepository.com/)

Rivest, R. L. (1994). *The RC5 Encryption Algorithm* (Tech. Rep. No. MIT/LCS/TR-595). MIT Laboratory for Computer Science. (Retrieved June 20, 2024, from https://people.csail.mit.edu/rivest/pubs/Riv95.pdf)

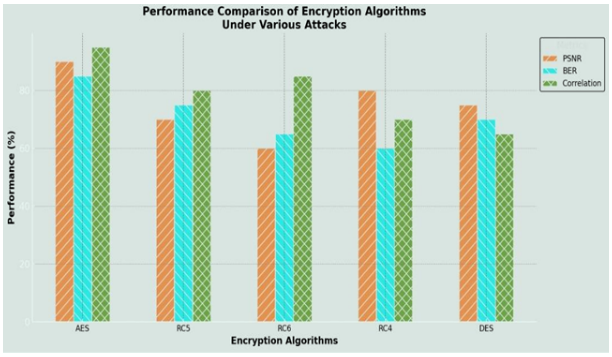
Sahu, R. (2023). *How cryptography works to secure our data: Understanding encryption, decryption, and hashing.* Medium. (Retrieved January 20, 2023 from https://medium.com/

@riteshs4hu/cryptography-1fc4d99e56cc)

Sethi, N., & Vijay, S. (2013). Comparative image encryption method analysis using new transformed-mapped technique. In *Proceedings of the conference on advances in communication and control systems 2013 (cac2s 2013)* (pp. 46–50). Dehradun, India: Atlantis Press. doi: 10.2991/cac2s.2013.12

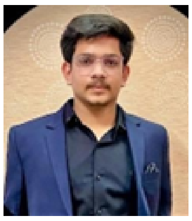
Zheng, P., & Zhang, Y. (2020). A robust image watermarking scheme in hybrid transform domains resisting rotation attacks. *Multimedia Tools and Applications*, *79*(7). doi: 10.1007/s11042-019-08490-4

Graphical Abstract



Sudhanshu Chaudhary is pursuing a Bachelor of Technology (B.Tech) in Data Science from J. C. Bose University of Science & Technology, YMCA, Faridabad. He has completed coursework in various fields, including machine learning, data mining, security algorithms, image processing, and statistical analysis, enhancing his analytical skills and technical knowledge in data science.

Samyak Jain is pursuing a Bachelor of Technology (B.Tech) in Information Technology from J. C. Bose University of Science & Technology, YMCA, Faridabad. He has engaged in a diverse curriculum that includes software development, database management, image processing and cybersecurity, allowing him to develop a strong foundation in IT principles and practices.



Pratham Jain is pursuing a Bachelor of Technology (B.Tech) in Computer Science from J. C. Bose University of Science & Technology, YMCA, Faridabad. His academic journey has provided him with a comprehensive understanding of computer programming, software engineering, image processing, and data structures, equipping him with the essential skills for a successful career in technology.