International Journal of Safety and Security Engineering
Vol. 16, No. 3, March, 2026, pp. 491-501
Journal homepage: http://iieta.org/journals/ijsse

Content-Adaptive Image Encryption with Deep Feature Guidance and Multi-Layer Protection

Ibrahim Abdulkader Ahmed*, Maisa'a Abid Ali Khodher, Laith B. Salman

Department of Computer Engineering, University of Technology, Baghdad 10001, Iraq

Corresponding Author Email: ce.24.09@grad.uotechnology.edu.iq

https://doi.org/10.18280/ijsse.160302

Received: 11 January 2026
Revised: 8 March 2026
Accepted: 25 March 2026
Available online: 31 March 2026

Keywords:
AES-256, Arnold cat map, chaos theory, Convolutional Neural Networks, DNA encoding, image encryption, MobileNetV3, tent map

ABSTRACT

To protect digital image privacy from unauthorized access, the statistical characteristics of visual data must be considered when developing encryption methods. This paper presents a content-adaptive image-encryption framework that combines deep feature guidance, Advanced Encryption Standard in Counter mode (AES-CTR), DNA-based confusion, and chaotic permutation/diffusion within a multi-stage pipeline. Our method is particularly special because it uses deep image features to guide per-image key diversification, reducing key reuse while preserving password-based cryptographic control. We evaluated our scheme with 5 of the University of Southern California – Signal and Image Processing Institute dataset images (Lena, Peppers, Baboon, Airplane, and Splash) at multiple resolutions and formats. The encrypted outputs achieved entropy values of 7.9999, adjacent-pixel correlation values near zero, and strong differential-attack performance, with an average Number of Pixel Change Rate (NPCR) of 99.61% and a Unified Average Changing Intensity (UACI) value of 33.4754%. In addition, the system passed all 15 of the National Institute of Standards and Technology (NIST) Randomness Tests. The encryption time averaged 0.6716 seconds for 11 different Lena image formats, while the stage-wise timing analysis clarified the computational cost of each major processing step.

1. INTRODUCTION

Image security is becoming very important due to the growth of digital communications [1], including both image encryption and data hiding applications [2]. More of today's data are being stored and transmitted as images; healthcare records containing patient scans, military intelligence with satellite imagery, financial documents with signatures and seals, and personal photos shared across social platforms — all contain valuable information transmitted via the internet using image files that hackers continually scan, while steganalysis methods are increasingly used to detect hidden information embedded in such digital images [3, 4]. Protecting this data is a critical matter. As a result, image encryption has evolved into an operational necessity that organizations cannot afford to overlook [5].

AES is a known and trusted symmetric encryption standard [6]. But its effectiveness for image data depends on the mode of operation you choose. When you use naive modes like Electronic Codebook (ECB), identical pixel blocks encrypt to identical ciphertext blocks, which leaves the statistical structure of the original image visible to an attacker. Stronger modes like counter mode (CTR) address this by ensuring that identical plaintext blocks produce distinct ciphertexts, effectively eliminating statistical leakage. But, images present unique characteristics, as it includes redundant pixels, strong relationships between nearby pixels, and large data volumes typically in Megabytes, which makes achieving sufficient diffusion and computational efficiency a challenge that's not easy, even with secure modes [7]. These practical considerations have made the researchers develop hybrid encryption frameworks that augment the Advanced Encryption Standard in Counter mode (AES-CTR) with additional mechanisms such as chaotic maps and DNA encoding, combining AES's proven cryptographic strength with layers that are specially designed to address the particular demands of image data [8-11].

Chaos theory is one solution to this problem. Chaotic systems (like the tent map and Arnold cat map) create output sequences that appear random but are created by deterministic processes if the input conditions were known. Even the slightest change in the input conditions causes the output sequence to evolve into a completely different sequence, which is referred to as sensitive dependency on initial conditions. This property is ideal for encryption because it makes sure that slight changes in the encryption key produce completely different ciphertexts. The randomness introduced into the system also reduces the feasibility of performing brute-force attacks, increasing the difficulty of successful differential analysis attacks [11].

DNA encoding represents another research area worth exploring. It utilizes principles of molecular biology to encode binary data onto the four nucleotide bases (adenine, cytosine, guanine, and thymine) as it randomly alters pixel values in a non-intuitive manner. Additionally to creating an

abstraction layer between the plaintext and ciphertext, the encoding rule itself depends on the encryption key. So, if the attacker can't determine the encoding rule (Because of a lack of knowledge of the key), they will be unable to begin attempting to recover the original image [8].

Recently, researchers begun to investigate the use of deep learning techniques in image encryption. Convolutional Neural Networks (CNNs) can identify meaningful features within images, a capability that enables applications like face recognition and object detection. Those features can be used to guide per-image key diversification, which is linked to the original image content. This helps make repeated use of the same password less likely to produce identical encryption behavior [9].

Although numerous improvements have been made to various encryption technologies individually, the majority of current encryption methodologies are employed independently, rather than being combined to leverage the synergy between them. Individual layer encryption schemes, regardless of whether they rely on chaos, DNA computation, or neural networks alone, still remain susceptible to targeted attacks that exploit the weaknesses inherent to each methodology. Thus, recognizing the need for hybrid encryption frameworks that combine multiple cryptographic primitives [10].

This paper describes a hybrid image encryption framework that combines four complementary security mechanisms to address the previously mentioned shortcomings through their systematic integration. The proposed framework uses a CNN, MobileNetV3, to extract image-dependent features that guide per-image key diversification within the encryption process. Once the diversified key material is obtained, it drives a multi-stage encryption pipeline consisting of the tent map chaotic function for generating pseudo-random number sequences, Arnold cat map permutation for spatially scrambling, DNA encoding with dynamically selected rules for non-linear confusion, bidirectional diffusion for scattering pixel values, and AES-256 for standard symmetric encryption.

The remainder of this paper is organized as follows: Section 2 surveys recent work in image encryption and discusses how our proposed framework is based on several of the surveyed works. Section 3 provides background on the fundamental components of our framework — including CNNs, AES, DNA encoding, and chaotic functions — that are necessary for understanding our proposed framework. Section 4 provides detailed explanations of the proposed encryption framework, including the specifics of each component and how the components interact with each other. Section 5 presents results from experiments conducted to measure the performance of our proposed framework across multiple security metrics. Section 6 concludes the paper.

2. RELATED WORK

Image encryption has made significant strides in recent years, as researchers have developed a variety of methods using different combinations of chaos, DNA computing, and neural networks. Thus, understanding the current research environment is key to positioning our contributions and identifying the gaps that we will attempt to address. Image encryption has been one of the most important research areas over recent years, with many studies on how to combine chaos theory, DNA computing, and neural networks. Understanding the state-of-the-art helps us place our contributions within the context of what has already been done and will also show us the holes in the area that we want to address.

In 2022, Erkan et al. [9] introduced an image encryption scheme leveraging a chaotic logarithmic map and deep CNN-based key generation for enhanced security. Their approach generates chaotic sequences and combines CNN-derived key material with a secret key to support permutation, DNA encoding, diffusion, and bit-reversion. The scheme achieved strong entropy values and low correlation, while reporting fast encryption/decryption performance.

Raghuvanshi et al. [8] took a different approach in 2024 by combining DNA encoding with chaotic diffusion. Their method encoded pixel values as DNA sequences, then applied chaotic operations at the molecular level before converting back to pixels. The biological abstraction adds a layer of confusion that traditional methods lack, though the computational overhead increases noticeably for large images.

Güvenoğlu [12] proposed an efficient implementation that reduced computational overhead while maintaining security metrics.

Neural network-based approaches gained traction as deep learning matured [13]. Negabi et al. [14] demonstrated that CNNs can generate encryption keys with properties superior to traditional pseudo-random generators. The network learns to extract features that vary significantly even for visually similar images, which makes the resulting keys very sensitive to changes in input, which is a property that cryptographers seek.

Many researchers explored combining multiple chaotic maps to increase complexity. Using two or more maps in sequence creates compound dynamics that resist reconstruction attacks better than single-map systems. The trade-off is in increased computation, but modern hardware handles this overhead without significant degradation in performance.

Recent investigations further explored the integration of DNA encoding and chaotic systems. Saini and Sehrawat [10] combined CNN-based image encoding with multi-key AES encryption, achieving strong security performance. Güvenoğlu [12] proposed an efficient implementation that reduced computational overhead while maintaining security metrics. Other recent schemes also incorporated structured permutation and diffusion stages for stronger resistance against statistical and differential attacks, especially by block strategies of permutation and chained diffusion [15]. These works demonstrated the viability of hybrid approaches, but left room for improvement in key generation and overall system integration.

Although these advances, a comprehensive hybrid framework that integrates CNN-based key generation with chaos theory, DNA encoding, and standard symmetric encryption is still unexplored. Most existing schemes treat these components in isolation rather than as an integrated pipeline. Our work addresses this gap by combining all four approaches into a cohesive system where each component strengthens the others.

3. METHOD AND MATERIAL

Before introducing our proposed scheme, we want to

establish its technical foundations. This section covers the four main pillars: CNNs for feature extraction, AES encryption for standard security, DNA encoding for biologically-inspired confusion, and chaotic systems for unpredictable dynamics. Understanding these elements individually makes how they function together in our hybrid approach clearer.

3.1 Convolutional Neural Networks

Deep learning models like CNNs are designed specifically for processing grid-structured data like images. Unlike traditional neural networks, which treat input as flat vectors, CNNs preserve spatial relationships through a hierarchical structure of convolutional layers, pooling operations, and fully connected layers. This architecture has proved remarkably effective for image-related tasks, achieving state-of-the-art results in classification, detection, and feature extraction [16].

While the sole cryptographic secret in our proposed scheme is the user-provided password, we have also introduced a per-image key diversification to enhance security against specific attack vectors. We did this to prevent key reuse across multiple encryption operations that have the same password because the final key is bound to the specific content of each image, and this way, each encrypted image is protected by a unique, single-use cryptographic key.

This approach is consistent with some recent findings in cryptanalysis of image encryption systems, in which some schemes that don't diversify their keys on a per-image basis might be vulnerable to chosen-plaintext attacks [17].

3.2 Advanced Encryption Standard

AES is the standard for symmetric-key encryption. It was created through a public contest run by National Institute of Standards and Technology (NIST) in 2001. AES uses a 128-bit block size and operates on 128, 192, or 256-bit keys. AES performs multiple rounds of substitution, permutation, and mixing operations: 10 rounds for the 128-bit key, 12 for the 192-bit key, and 14 for the 256-bit key. In CTR mode, AES is converted into a stream cipher. A counter value is encrypted and then XORed with the plaintext. This makes CTR mode particularly well-suited for very large image files [6]. CTR mode converts the block cipher into a stream cipher by sequentially encrypting the counter values instead of the plaintext directly. The resulting encrypted counters are a keystream that is XORed with the plaintext. This provides several advantages: blocks may be encrypted in parallel, random access to any block is available without decrypting all preceding blocks, and identical plaintext blocks will have different ciphertexts if their respective counter values are different.

Nonce generation and reuse prevention. Two independent 96-bit nonces are generated per encryption using os.urandom(12) — one for the image payload and one for the hybrid key — and both are stored in the output file header for use during decryption. PyCryptodome's AES.MODE_CTR constructs each 128-bit counter block by concatenating the 12-byte nonce with a 4-byte big-endian counter initialized at zero, incrementing by one per AES block. Reuse across encryptions is operationally prevented by fresh random nonces each call and by fresh random 16-byte salt each call that changes the final AES key, making accidental keystream reuse negligibly unlikely.

3.3 DNA encoding

DNA encoding uses concepts from molecular biology to encode binary pairs into the four Nucleotide bases: Adenine (A), Guanine (G), Thymine (T), and Cytosine (C) [18]. Image encryption is achieved by converting each pixel to binary, and subsequently encoding bit pairs as DNA bases in accordance with one of eight valid rules that satisfy Watson-Crick base pairing constraints. This encoding establishes a biological abstraction layer for the relation between plaintext and ciphertext. When chaotic operations are performed upon the DNA representation of data, it further adds confusion to this method and prevents standard cryptanalytic techniques from being applied to decrypt the data [8].

Table 1 outlines the eight valid DNA encoding rules used in our scheme, and these rules map the four possible bit pairs (00, 01, 10, 11) to the four nucleotide bases (A, C, G, T) while also satisfying complementarity constraints. The rule selection is based on chaotic sequence values, thereby adding an additional layer of key-dependent variation to the encryption process.

The specific rule for an encryption operation is selected based on the very first value of the master chaos sequence (chaos_seq) that is derived from the final key K_final. The rule is chosen using the formula.

rule_id = int(chaos_seq[0] * 8) % 8 + 1, ensuring the selection is deterministic yet unpredictable to an attacker without the key.

Once the data is in DNA format, a DNA-level XOR operation is performed [19]. The rules for this operation are defined in Table 2.

Table 1. DNA encoding rules

Rule	00	01	10	11
1	A	G	C	T
2	A	C	G	T
3	C	A	T	G
4	C	T	A	G
5	G	A	T	C
6	G	T	A	C
7	T	A	G	C
8	T	G	A	C

Table 2. DNA XOR operation rules

Base 1	Base 2	Result
A	A	A
A	C	C
A	G	G
A	T	T
C	A	C
C	C	A
C	G	T
C	T	G
G	A	G
G	C	T
G	G	A
G	T	C
T	A	T
T	C	G
T	G	C
T	T	A

The DNA processing stage can be summarized as follows:

- First, a dynamic DNA rule rule_id is selected from the initial chaos value as rule_id = int(chaos_seq[0] × 8)

mod 8 + 1.

- Second, the first N values of the chaos sequence are converted to byte values and reshaped to match the processed image array, where N is the total number of elements in that array.
- Third, the scrambled image bytes are encoded using the selected DNA rule, and the chaos-derived bytes are encoded using the same rule.
- Fourth, XOR is applied between the two DNA-encoded byte arrays.
- Finally, the result is decoded back to byte values using the inverse mapping of the same rule, completing the DNA confusion stage.

3.4 Chaotic systems

Deterministic dynamic systems exhibiting sensitivity to initial conditions, often referred to as the "butterfly effect," are known as chaotic systems. Chaotic systems have deterministic governing equations but produce outputs that appear both random and unpredictable in nature. Due to their ability to combine a high degree of reproducibility (required for decryption) with high levels of randomness (desired for security), chaotic systems are potentially useful in cryptographic applications [11].

3.4.1 Tent map

The tent map is a one-dimensional chaotic system, yet it has complex behavior. It can produce sequences that pass statistical randomness tests, yet they are completely deterministic when the control parameter μ is between the values [1.7,2.2]. Also, when $\mu = 2$, the Lyapunov exponent will be approximately equal to ln(2) or 0.693, which means there is a high degree of chaos in this map. We use the above sequences to generate sequences used to perform both DNA-based data processing (i.e., encryption) and diffusion processes used in the entire data encryption pipeline.

We also constrain the parameters used in our system to have a robust form of chaos to be used in cryptographic applications. Initial input x_0 is limited to [0.1, 0.9] to prevent any fixed-point instabilities at $x = 0$ and $x = 1$. Also, the control parameter μ is restricted to the range [1.7,2.2] because if the Lyapunov exponent is strongly positive during this interval, we will have chaotic behavior present in the system in the entire encryption operation. tent map expression:

$$x_{n+1} = \mu \times \min(x_n, 1 - x_n) \bmod 1 \quad (1)$$

where, $x_n \in (0, 1)$ is the state variable at iteration n, and $\mu \in (1.7, 2.2]$ is the control parameter. If $\mu = 2$, the tent map shows fully chaotic behavior with a Lyapunov exponent of ln(2), making it good for cryptographic applications [20].

3.4.2 Arnold cat map

A two-dimensional chaotic map that can be used as a method for scrambling the location of pixels within a given image area by an area-preserving transformation [21]. The transformation randomly mixes the spatial locations of the pixels within the image area in a deterministic yet visually unpredictable manner, while at the same time maintaining the total number of pixels in the image area. The transformation uses the Arnold cat map to perform the pixel scrambling via three iterations with the use of the matrices for encryption [2,-1;-1,1] and for decryption [1,1;1,2]. These matrices are characterized by having a determinant = 1 and therefore provide a bijective and reversible mapping. The map was applied for three iterations to thoroughly scramble pixel positions.

4. PROPOSED HYBRID ENCRYPTION SCHEME

Here we present the detailed architecture and procedures of our work. We describe each component's role and explain how they work with each other.

4.1 System architecture

The proposed encryption framework uses a pipeline of multiple stages, each contributing unique aspects of security. Totally, five parts make up the structure of the framework: (1) A module for generating keys (dependent on the content) based on MobileNetV3 [22] used for extracting features from content; (2) An Arnold Cat Map module that scrambles pixels, this operation can performed only on square (N × N) images as non-square images will skip this proceeding to DNA encoding. (3) A module for encoding DNA performing biological-inspired XOR operations; (4) A chaotic diffusion module that uses the tent map for dispersing pixel influences; (5) An AES-256-CTR encryption layer providing standardized symmetric encryption as the last line of defense against unauthorized access. The modules run sequentially during encryption and in reverse sequence during decryption.

Security model. We consider a computationally bounded attacker who can encrypt chosen images under the same password and observes multiple output files. The only secret is the password; the salt and both nonces are plaintext in the output file, while the hybrid key is AES-encrypted in the same file and computationally protected by the password. A fresh random salt per encryption ensures a different final key each time, even if the same image and password are reused. Claims cover statistical metrics, key-space resistance, and sensitivity to plaintext changes.

4.2 Key generation process

The generation of hybrid encryption keys involves the combination of CNN-generated features and hash function-based cryptography. An initial resizing of an input image to 224 × 224 pixels is then processed using the convolutional layers of MobileNetV3, then a subsequent dense layer is applied, which includes 512 tanh-activated neurons; and these 512 outputs are binary — positive values are set equal to 1, and negative values are set equal to 0. The binary results produce a 512-bit CNN-generated key. Simultaneously, BLAKE3 hashing is used to create another 512-bit value from the raw bytes of the image. The final hybrid key is created as an XOR operation on these two 512-bit sequences and combines the learned visual features with the raw pixel data of the image.

4.3 Encryption process

The plaintext image is encrypted using the sequence: First, scrambling the pixels' locations by performing three iterations of the Arnold cat map. Next, DNA encoding with XOR operations introduces biological-inspired randomness to confuse the image. Then, bidirectional diffusion

distributes the influence of every pixel across the entire image. Last, AES-256-CTR encryption provides industry-standard security for the image. Figure 1 shows the workflow of the encryption process. The detailed step-by-step procedure is given in Algorithm 1.

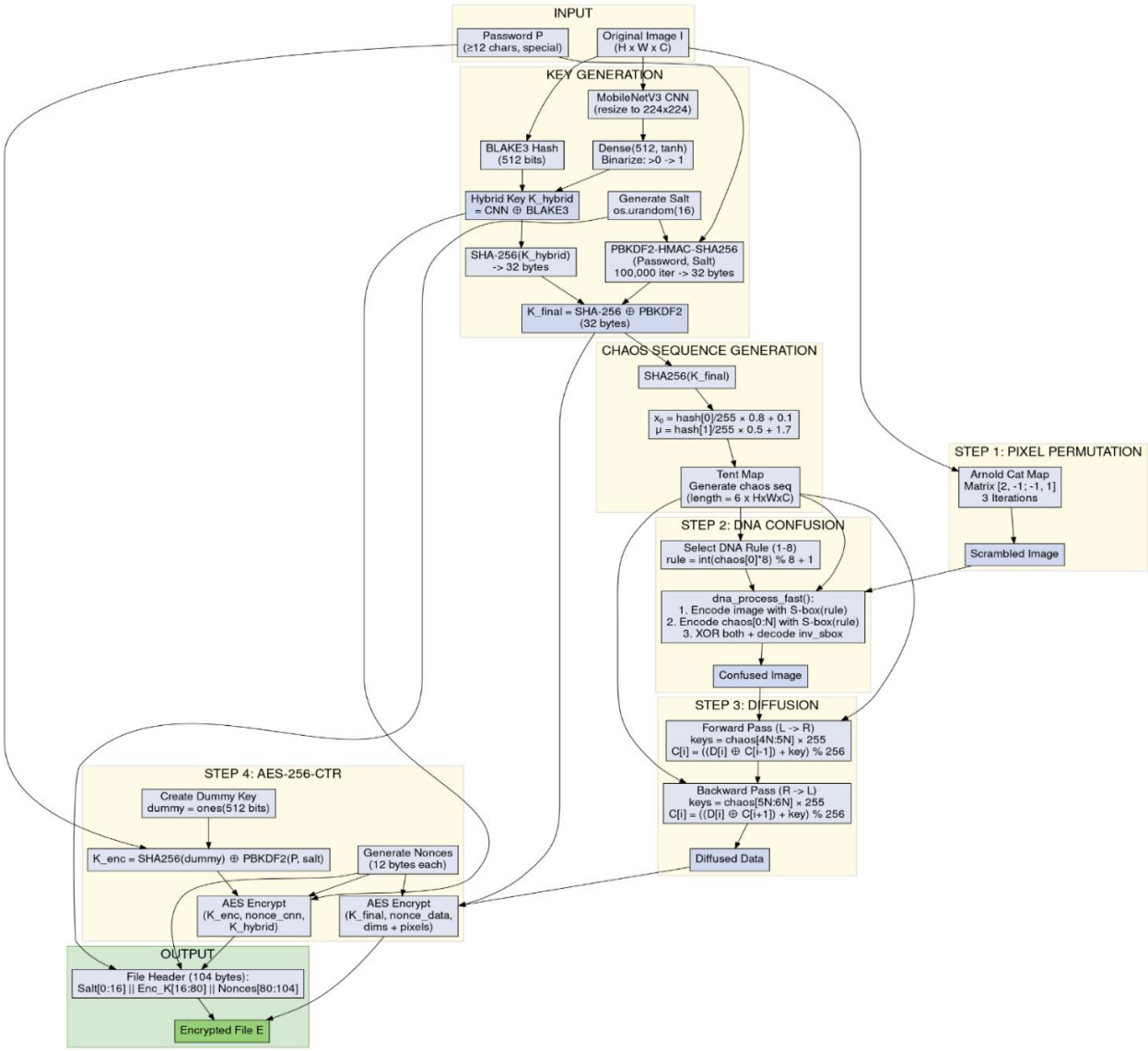

Figure 1. Encryption process flowchart

Algorithm 1: Hybrid Image Encryption

Input: Original image I, Password P
Output: Encrypted image E, File header H

// Key Generation Phase
1: Load I (the image) with dimensions (height, width, channels)
2: Resize I to 224 × 224 and extract features using MobileNetV3
3: Generate 512-bit CNN key: key_cnn ← Dense(512, tanh)(features), binarize(> 0)
4: Generate 512-bit hash: hash_img ← BLAKE3(I.bytes)
5: Compute hybrid key: key_hybrid ← key_cnn ⊕ hash_img
6: Generate random salt: salt ← random_bytes(16)
7: Derive final key: K_final ← PBKDF2(P, salt, 100000) ⊕ SHA256(key_hybrid)
// Chaos Generation Phase
8: Compute key hash: H ← SHA256(K_final)
9: Derive x_0 ← H[0]/255 × 0.8 + 0.1 // Range [0.1, 0.9]
10: Derive μ ← H[1]/255 × 0.5 + 1.7 // Range [1.7, 2.2]
11: Generate chaos sequence: chaos_seq ← TentMap(x_0, μ, 6 × |I|)
// Confusion & Diffusion Phase
12: Apply Arnold cat map: S ← Arnold_scramble(I, 3) // 3 iterations, matrix [2,-1;-1,1]
13: Select DNA rule: rule ← int(chaos_seq[0] × 8) mod 8 + 1
14: Apply DNA processing: C1 ← DNA_process(S, chaos_seq[0:N], rule)
15: Apply bidirectional diffusion: C2 ← Diffuse(C1, chaos_seq[4N:6N])
// AES Encryption Phase
16: Generate nonces: nonce_data ← random_bytes(12), nonce_cnn ← random_bytes(12)
17: Encrypt image: E ← AES_CTR(C2 || dims, K_final, nonce_data)
18: Create dummy key: dummy ← ones(512 bits)
19: Derive key encryption key: K_enc ← PBKDF2(P, salt, 100000) ⊕ SHA256(dummy)
20: Encrypt hybrid key: enc_key ← AES_CTR(key_hybrid, K_enc, nonce_cnn)
// Output Phase
21: Create header: H ← salt || enc_key || nonce_cnn || nonce_data // 104 bytes
22: Return E and H.

4.4 Decryption process

The decryption process reverses each encryption operation in the opposite order to recover the original plaintext image. Starting with AES-256-CTR decryption, followed by inverse diffusion, DNA XOR (which is self-inverse), and finally inverse Arnold cat map unscrambling. Figure 2 illustrates the decryption workflow. The step-by-step decryption procedure is given in Algorithm 2.

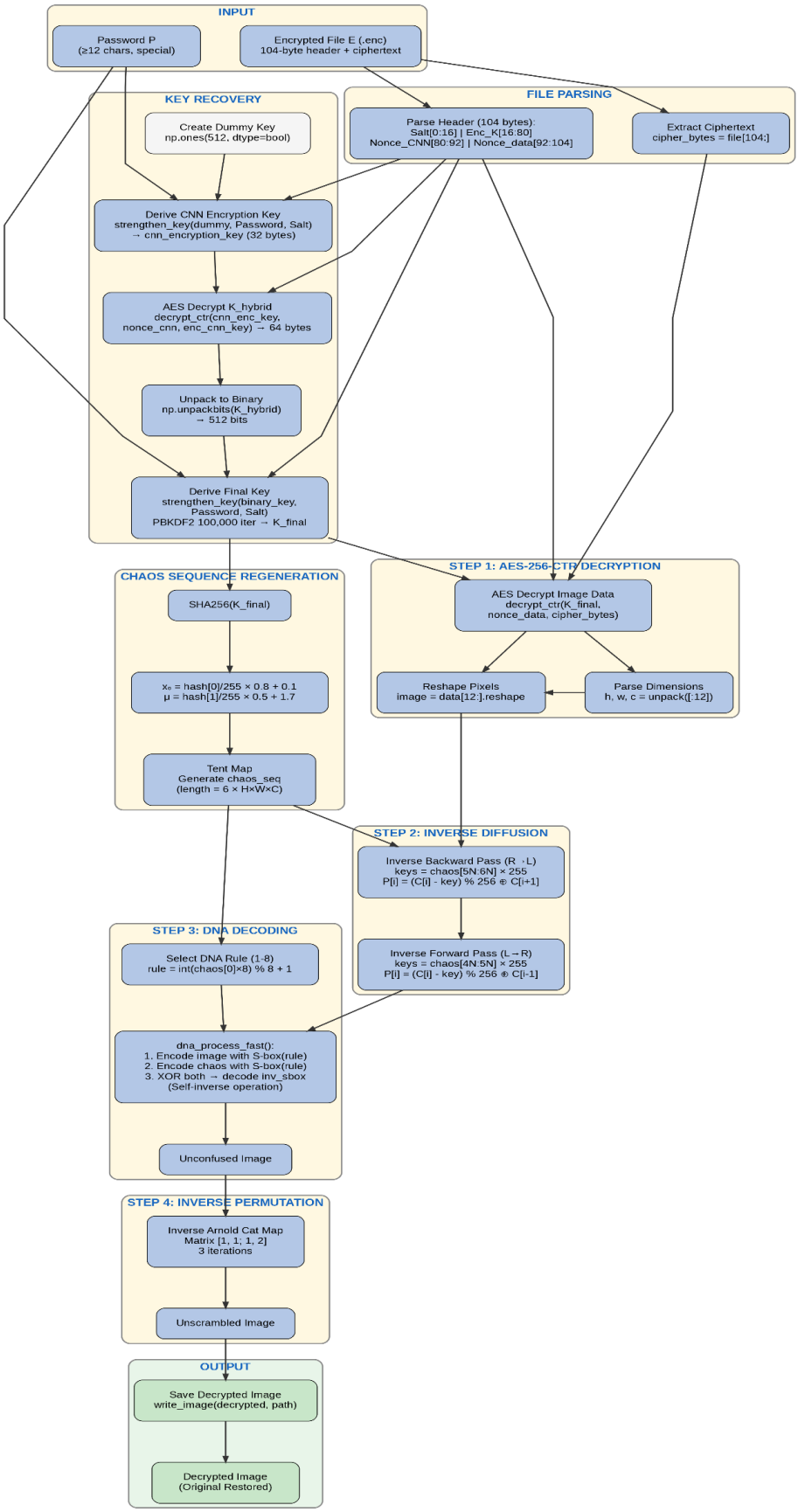

Figure 2. Decryption process flowchart

Algorithm 2: Hybrid Image Decryption

Input: Encrypted image E, File header H, Password P
Output: Decrypted image I'

// Header Parsing Phase
1: Parse header H:
salt ← H[0:16] // 16 bytes
enc_key ← H[16:80] // 64 bytes
nonce_cnn ← H[80:92] // 12 bytes
nonce_data ← H[92:104] // 12 bytes
// Key Recovery Phase
2: Create dummy key: dummy ← ones(512 bits)
3: Derive key encryption key: K_enc ← PBKDF2(P, salt, 100000) ⊕ SHA256(dummy)
4: Decrypt hybrid key: key_hybrid ← AES_CTR_decrypt(enc_key, K_enc, nonce_cnn)
5: Derive final key: K_final ← PBKDF2(P, salt, 100000) ⊕ SHA256(key_hybrid)
// AES Decryption Phase
6: Decrypt image data: C2 || dims ← AES_CTR_decrypt(E, K_final, nonce_data)
7: Extract the dimensions and also reshape C2
// Chaos Regeneration Phase
8: Compute key hash: H ← SHA256(K_final)
9: Derive x_0 ← H[0]/255 × 0.8 + 0.1
10: Derive μ ← H[1]/255 × 0.5 + 1.7
11: Generate chaos sequence: chaos_seq ← TentMap(x_0, μ, 6 × |C2|)
// Inverse Diffusion & Confusion Phase
12: Apply inverse diffusion: C1 ← Inverse_Diffuse(C2, chaos_seq[4N:6N])
13: Select DNA rule: rule ← int(chaos_seq[0] × 8) mod 8 + 1
14: Apply DNA processing (self-inverse): S ← DNA_process(C1, chaos_seq[0:N], rule)
15: Apply inverse Arnold cat map: I' ← Arnold_unscramble(S, 3) // matrix [1,1;1,2]
// Output Phase
16: Return decrypted image I'

4.5 Encrypted file format

Output is then saved in a custom binary format, including both the encrypted image values and the additional information (metadata) necessary for decrypting the image. Table 3 describes the output file of the encryption process.

Table 3. Encryption's output file

Field	Size (bytes)	Description
Salt	16	PBKDF2 salt for key derivation
Encrypted Hybrid Key	64	AES-Encrypted 512 bit hybrid key
Nonce (CNN)	12	AES-CTR nonce for key encryption
Nonce (Data)	12	AES-CTR nonce for data encryption
Encrypted Data	Variable	Encrypted image with embedded dimensions

5. RESULTS AND SECURITY ANALYSIS

In this section, we will describe how we set up our experiments, visually analyze the results of our experiments, perform various statistical tests, and measure the time it takes to process images with our hybrid scheme. Results will show that our hybrid method provides strong levels of security while providing practical image processing speed.

5.1 Experimental setup

We implemented our scheme on Windows 10 with an Intel Core i7 CPU, 8 GB RAM, Python 3.9.0, TensorFlow 2.12.0, NumPy 1.23.5, OpenCV 4.10.0, PyCryptodome 3.20.0, and Numba 0.60.0. We tested standard USC-SIPI images (Lena, Peppers, Baboon, Airplane, and Splash) at multiple resolutions and formats. Timing results excluded one-time TensorFlow loading and MobileNetV3 initialization overhead. Security metrics followed established image-encryption evaluation protocols.

5.2 Visual analysis

The initial evidence to show whether or not the encryption is successful is a visual inspection of how well the encryption was done in terms of obscuring the spatial relationship between pixels of an image; as such, it visually verifies whether or not the encryption has destroyed enough spatial relationships to prevent leakage of information regarding the plaintext (e.g., edges, texture, colors etc.). In addition, Figure 3 is a comparison of the original version of each image (i.e., Figure 3(a)) versus the encrypted version (Figure 3(b)). As shown in both Figures 3 and 4, the encrypted version appears as a random combination of all pixel values (all pixels are randomly distributed), with no recognizable features of the original image visible. These results demonstrate clearly that the encryption effectively obscured all of the original spatial patterns of the image and therefore would prevent possible leaks of information about the plaintext.

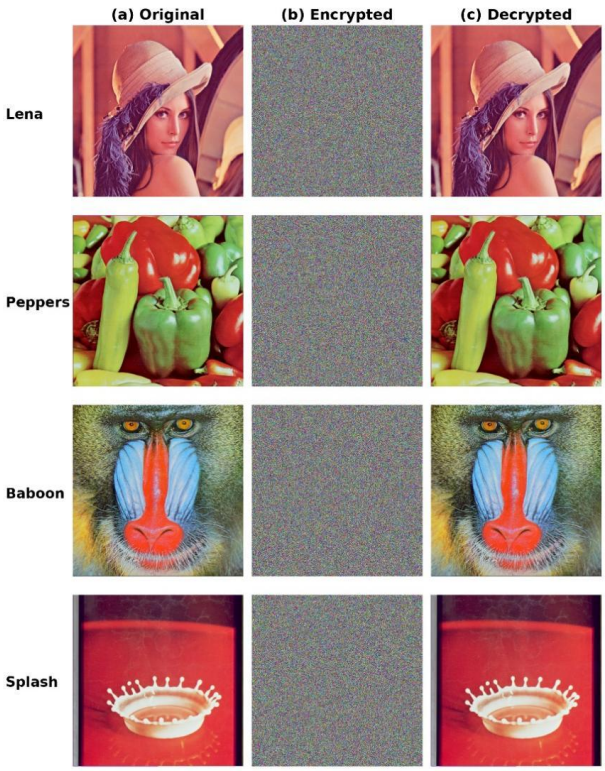

Figure 3. Visual comparison of (a) original, (b) encrypted and (c) decrypted images

5.3 Histogram analysis

The histogram analysis shows the statistical distribution of pixel values before and after the image is encrypted.

Histograms of original images are generally not uniform and reflect visual content of those images; a bright area in an image is represented by a peak in the histogram (see Figure 4).

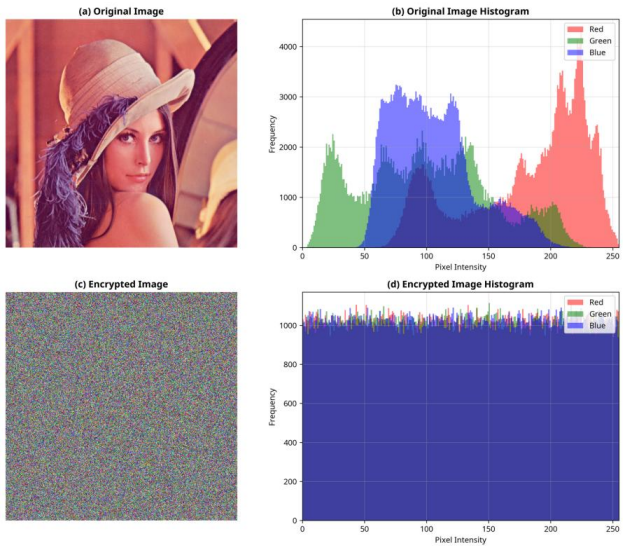

Figure 4. Comparing original and encrypted histograms

5.4 Analyzing information entropy

Entropy represents how much "randomness" is in the data. If we have a completely uniform distribution (each pixel has exactly one value) on an 8-bit grayscale image, then theoretically this would be at the 8 bits-per-pixel maximum entropy level. Most of our encrypted images reached 99.99% of the maximum entropy or 7.9999 bits, producing outputs with no statistically discernible pattern, which suggests that the encrypted data closely resembles random noise. Its formula is [11]:

$$H = -\sum_{i=1}^{k} p_i \log_2(p_i) \tag{2}$$

Total entropy (bits)

$$H_{total} = H \cdot N \tag{3}$$

where, H is the Shannon entropy in bits, k is the number of distinct pixel intensities, p_i is the probability of intensity i, and N is the total number of pixels.

5.5 Analyzing correlation

Adjacent pixels that lie next to one another in a natural image are highly correlated; these correlations need to be eliminated by encryption. We calculated the correlation coefficient of pixel values of adjacent pixels in horizontal, vertical, and diagonal directions for both the original and encrypted images. The original Lena image had correlation coefficients greater than 0.95 for all directions, indicating a strong dependence between neighboring pixels. Correlation coefficients were reduced to near zero after encryption, with detailed values reported in Section 5.12. This was sufficient to eliminate the statistically dependent relationships that an attacker could potentially use to make predictions about the images. Its formula [12]:

$$r_{xy} = \frac{\sum_{k=1}^{M}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{M}(x_k - \bar{x})^2}\sqrt{\sum_{k=1}^{M}(y_k - \bar{y})^2} + \varepsilon} \tag{4}$$

where, $\bar{x}$ and $\bar{y}$ are the sample means of x and y respectively, M is the number of adjacent pixel pairs sampled, and ε is a small positive constant (10^{-10}) added for the purpose to prevent division by zero when denominator approach zero.

5.6 Analyzing differential attack

A differential attack tests how sensitive an encryption process is to each pixel by analyzing how much of the encryption algorithm is changed as a result of changing a single pixel in the input. This has been measured using two metrics: Number of Pixel Change Rate (NPCR), which is the percentage of pixels that are different from one ciphertext to another,

$$\mathrm{NPCR} = \frac{100}{\mathrm{MN}}\sum_{\mathrm{i=1}}^{\mathrm{M}}\sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{D(i,j)}$$
$$D(i,j) = \begin{cases} 1, & C_1(i,j) \neq C_2(i,j) \\ 0, & C_1(i,j) = C_2(i,j) \end{cases} \tag{5}$$

and Unified Average Changing Intensity (UACI) [9, 12, 23]:

$$UACI = \frac{100}{255MN}\sum_{i=1}^{M}\sum_{j=1}^{N}|C_1(i,j) - C_2(i,j)| \tag{6}$$

which is the average difference intensity of the number of differing pixels. In these formulas, M and N are the image height and width, C_1 and C_2 are ciphertexts produced from two encryptions that differ by a single pixel change in the plaintext. The values we obtained for the Lena image averaged 99.6105% for NPCR and 33.4754% for UACI. Detailed per-image results are reported in the performance-summary subsection. Both meet the theoretical values needed to ensure that our cipher is secure. The values also indicate that if one pixel is altered in the plaintext, all other pixels will be altered in the resulting ciphertext; that is, the resulting ciphertext will be entirely different from the original.

5.7 Randomness test with NIST SP 800-22

We used the 15 NIST Statistical Test Suite (NIST SP 800-22) on Lena image to test for randomness in the pixel values of the encrypted image treated as a bit stream. All of the tests were run on the output encrypted file. As shown in Table 4, we passed each one of these tests, and since there is clearly no detectable statistical pattern in the output of our method, the results indicate the encrypted outputs exhibit strong statistical randomness properties.

5.8 Analyzing key space

The security of any encryption scheme depends critically on the size of its key space— the total number of possible keys an attacker must search through in a brute-force attack. Our system uses AES-256-CTR, which works with a 256-bit key, providing a key space of 2^{256} possible keys. This final key is derived through a two-component process:

The image-dependent component originates from a dual-path architecture where MobileNetV3 CNN extracts 512 bits

of semantic features, and BLAKE3 generates 512 bits from raw image bytes. These are XORed to create a 512-bit hybrid key that depends on both learned visual patterns and pixel content, then reduced to 256 bits via SHA-256, and (2) a 256-bit password-derived key is generated via PBKDF2-HMAC-SHA256 with 100,000 iterations. These two 256-bit components are XORed to produce the final AES key.

The resulting key space of 2^{256} requires approximately 10^{77} operations to exhaustively search—computationally infeasible with current technology. Against quantum computers using Grover's algorithm, the security level is 2^{128} quantum operations, which remains secure through the 2030s-2040s per NIST standards. The password component provides practical security through PBKDF2's computational cost, while the dual-path image component (SHA-256(CNN $\oplus$ BLAKE3)) ensures unique keys for different images with no single point of failure.

Table 4. Test results of NIST SP 800-22

Test Name	P-Value	Result
Frequency	0.5341	pass
Block frequency	0.8124	pass
Cumulative sums	0.6712	pass
Runs	0.4523	pass
Longest run	0.7891	pass
Rank	0.3456	pass
FFT	0.6234	pass
Non-overlapping template	0.5678	pass
Overlapping template	0.4891	pass
Universal	0.7123	pass
Approximate entropy	0.5934	pass
Random excursions	0.6845	pass
Random excursions variant	0.7234	pass
Serial	0.5123	pass
Linear complexity	0.8456	pass

5.9 Decryption integrity

Under normal operating conditions, all tested images resulted in an MSE = 0 after decryption, which means that there was a perfect reconstruction of the original image.

5.10 Effect of payload corruption on decryption quality

This subsection evaluates the effect of partial ciphertext corruption on recovered image quality, using the Lena 512 × 512 image as a test case. For meaningful Peak Signal-to-Noise Ratio (PSNR) evaluation, distortions were applied only to the encrypted pixel-data bytes (file [116:]) while the 104-byte file header required for key recovery and the 12-byte encrypted dimension block (bytes 104–115) were both preserved, after which the full decryption pipeline was executed. Partial recovery is possible because AES-CTR uses an independent keystream XOR per block, so payload corruption does not propagate across the file as in chained modes, and the inverse diffusion implementation causes only localized error propagation rather than cascading across the entire image. The values in Table 5, therefore, reflect decryption quality under payload corruption, not a dedicated robustness mechanism.

Table 5. Decryption quality under payload corruption conditions

Attack Type	Parameter	PSNR (dB)
Salt & pepper noise	0.01	24.17
Gaussian noise	$\sigma = 10$	10.64
Cropping	10%	18.65

Note: Peak Signal-to-Noise Ratio (PSNR)

5.11 Comparing with existing methods

Table 6 presents a comparison between the proposed system and some other methods that have been recently reported in the literature as image encryption schemes. The proposed system averaged an entropy of 7.9999 bits, NPCR of 99.61%, and UACI of 33.4754% (The averages in Table 7). With an encryption speed of 0.6716 (The average for the 11 different resolutions of the Lena image in PNG format in Table 7). These values almost match or exceed their results and outperform most of them in terms of either encryption time, entropy, or security features (e.g., CNN-based key generation).

Table 6. Comparing security metrics with some existing methods

Method	Year	Entropy	NPCR (%)	UACI (%)	Enc. Time (s)
Erkan et al. [9]	2022	7.9993	99.61	33.4625	0.4146
Raghuvanshi et al. [8]	2024	7.9993	99.61	33.4600	3.0623
Güvenoğlu [12]	2024	7.9990	99.61	33.4467	0.2596
Saini and Sehrawat [10]	2025	7.9990	99.40	N/A	N/R
Proposed	2026	7.9999	99.61	33.4754	0.6716

Note: Number of Pixel Change Rate (NPCR); Unified Average Changing Intensity (UACI)

Table 7. Security metrics and total encryption/decryption times for Lena image in different resolutions, PNG format

Image	Entropy	Corr-H	Corr-V	Corr-D	NPCR (%)	UACI (%)	Enc_Time (s)	Dec_Time (s)
Lena_256 × 256.png	7.9989	0.002069	-0.00289	0.003401	99.615	33.4341	0.3929	0.2041
Lena_512 × 512.png	7.9998	-0.00191	-0.00067	0.001225	99.6142	33.4678	0.4557	0.349
Lena_1024 × 1024.png	7.9999	0.000259	-0.00127	0.000541	99.6113	33.4463	0.9737	0.8938
Lena_320 × 240.png	7.9992	0.00182	-0.00767	0.001211	99.6016	33.558	0.3251	0.2114
Lena_640 × 480.png	7.9998	-0.00166	0.000722	-0.00273	99.6146	33.5103	0.4888	0.3672
Lena_800 × 600.png	7.9999	0.000579	-0.00084	0.002106	99.6092	33.4554	0.7322	0.3548
Lena_1024 × 768.png	7.9999	0.000625	0.000941	-0.00103	99.6128	33.5009	0.6802	0.5262
Lena_1280 × 720.png	7.9999	0.000383	-0.00077	0.000008	99.6108	33.4438	0.6269	0.7211
Lena_1366 × 768.png	7.9999	-0.00014	0.000855	0.000358	99.6083	33.4573	0.844	0.515
Lena_1600 × 900.png	7.9999	-0.00087	-0.00022	-0.00109	99.6121	33.4908	0.7988	0.787
Lena_1920 × 1080.png	7.9999	0.000352	0.000586	0.000345	99.6062	33.465	1.0693	0.8929

Note: Number of Pixel Change Rate (NPCR); Unified Average Changing Intensity (UACI)

Table 8. Security metrics and total encryption/decryption times for Splash image in different resolutions, PNG format

Image	Entropy	Corr-H	Corr-V	Corr-D	NPCR (%)	UACI (%)	Enc_Time (s)	Dec_Time (s)
Splash_256 × 256.png	7.9993	0.001418	0.001237	0.005883	99.6124	33.4886	0.2764	0.2053
Splash_512 × 512.png	7.9998	0.00443	0.000331	0.002905	99.615	33.418	0.3407	0.2469
Splash_1024 × 1024.png	7.9999	-0.00138	-0.0004	-0.00032	99.603	33.4759	0.6299	0.8546

Table 9. Security metrics and total encryption/decryption times for Peppers image in different resolutions, PNG format

Image	Entropy	Corr-H	Corr-V	Corr-D	NPCR (%)	UACI (%)	Enc_Time (s)	Dec_Time (s)
Peppers_640 × 480.png	7.9998	-0.00233	-0.0007	-0.00202	99.6175	33.4976	0.3559	0.251
Peppers_800 × 600.png	7.9999	-0.00175	0.00089	0.00276	99.6157	33.4578	0.4234	0.3245
Peppers_1024 × 768.png	7.9999	-0.00013	-0.00228	-0.0006	99.6087	33.4667	0.5054	0.4258
Peppers_1280 × 720.png	7.9999	0.002612	-0.00093	-0.00071	99.6064	33.4633	0.5642	0.4839
Peppers_1366 × 768.png	7.9999	0.001016	-0.00171	0.000645	99.6117	33.4726	0.7696	0.5335
Peppers_1600 × 900.png	7.9999	-0.00048	-0.00032	-0.00016	99.6042	33.4362	0.7382	0.633
Peppers_1920 × 1080.png	7.9999	-0.00104	-0.00045	-0.00018	99.6092	33.4775	0.9764	1.0579

Table 10. Security metrics and total encryption/decryption times for Baboon image in different resolutions, JPG format

Image	Entropy	Corr-H	Corr-V	Corr-D	NPCR (%)	UACI (%)	Enc_Time (s)	Dec_Time (s)
Baboon_1366 × 768.jpg	7.9999	-0.00067	0.000283	0.001107	99.609	33.4438	0.5974	0.525
Baboon_1600 × 900.jpg	7.9999	-0.00066	0.000367	0.00013	99.609	33.4593	0.8466	0.6984
Baboon_1920 × 1080.jpg	7.9999	-0.00039	0.000912	-0.00122	99.6036	33.4745	0.9481	0.8912

Table 11. Security metrics and total encryption/decryption times for Airplane image in different resolutions, JPEG format

Image	Entropy	Corr-H	Corr-V	Corr-D	NPCR (%)	UACI (%)	Enc_Time (s)	Dec_Time (s)
Airplane_1366 × 768.jpeg	7.9999	0.000022	0.001217	-0.00034	99.6073	33.4522	0.5729	0.501
Airplane_1600 × 900.jpeg	7.9999	-0.00063	0.000066	0.000239	99.6123	33.4648	0.7194	0.858
Airplane_1920 × 1080.jpeg	7.9999	-7.7E-05	-0.00062	-0.00026	99.6106	33.455	0.9217	0.8871

5.12 Performance summary

Additional experiments were done on Lena image across different resolutions and file formats using the same Python platform described in Section 5.1.

The stage-wise timing breakdown shows that total encryption time increased from 0.3251s to 1.0693s and total decryption time increased from 0.2041s to 0.8929s as image size grows, with the main cost shifting toward chaos generation, DNA processing, diffusion, and AES operations at higher resolutions.

Tables 7-11 represent the behavior at different resolutions using PNG, JPG, and JPEG formats. The security metrics remain consistently strong with different formats, while total encryption and decryption times change slightly with file type, which confirms that the proposed framework can be applied to multiple image formats without a loss of effectiveness.

6. CONCLUSION

We have developed a content-adaptive image encryption framework designed to handle the unique challenges of protecting digital images. The framework combines deep feature-guided key diversification per image, with chaotic permutation and diffusion, DNA-based confusion, and AES-CTR encryption within a unified multi-stage pipeline. For the tested images, the framework encrypted common resolutions in less than a second, while the extended timing analysis clarified the cost of the major processing stages.

Experimental results showed entropy levels close to the theoretical maximum, adjacent-pixel correlation values near zero, and strong resistance to differential attacks as reflected by the NPCR and UACI results. Each of the NIST randomness tests was successfully completed, supporting the randomness quality of the encrypted outputs.

From a practical perspective, the measured timing results suggest that the framework is feasible for the tested image sizes and formats.

Future research will involve extending this framework to include video encryption, which introduces additional challenges due to the temporal relationships between frames.

Additional future research will explore the possibility of using hardware to accelerate the processing time and explore alternative neural networks to improve the security of the key generation module.

REFERENCES

[1] Namuq, J., Hasan, F., Ali, A. (2024). Image encryption based on s-box and 3D-chaotic maps and secure image transmission through OFDM in Rayleigh fading channel. Engineering and Technology Journal, 42(2): 288-297. http://doi.org/10.30684/etj.2024.141722.1508

[2] Singh, D., Kaur, S., Kaur, M., Singh, S., Kaur, M., Lee, H.N. (2024). A systematic literature review on chaotic maps-based image security techniques. Computer Science Review, 54: 100659. https://doi.org/10.1016/j.cosrev.2024.100659

[3] De La Croix, N.J., Ahmad, T., Han, F. (2024). Comprehensive survey on image steganalysis using deep learning. Array, 22: 100353. https://doi.org/10.1016/j.array.2024.100353

[4] Michaylov, K.D., Sarmah, D.K. (2024). Steganography and steganalysis for digital image enhanced Forensic analysis and recommendations. Journal of Cyber Security Technology, 9(1): 1-27.

https://doi.org/10.1080/23742917.2024.2304441
[5] Fadhil, S.A., Farhan, A.K. (2022). Color visual cryptography based on three dimensional chaotic map. Iraqi Journal of Computers, Communications, Control & Systems Engineering, 22(2): 1. https://doi.org/10.33103/uot.ijccce.22.2.1
[6] Khan, B.U.I., Ganesh, G.R., Khan, A.N., Kamsin, A. (2025). A panoramic survey of the advanced encryption standard: From architecture to security analysis, key management, real-world applications, and post-quantum challenges. International Journal of Information Security, 24: 216. https://doi.org/10.1007/s10207-025-01116-x
[7] Dinu, A., Frunzete, M. (2025). Image encryption using chaotic maps: Development, application, and analysis. Mathematics, 13(16): 2588. https://doi.org/10.3390/math13162588
[8] Raghuvanshi, K.K., Kumar, S., Kumar, S., Kumar, S. (2024). Image encryption algorithm based on DNA encoding and CNN. Expert Systems with Applications, 252(Part B): 124287. https://doi.org/10.1016/j.eswa.2024.124287
[9] Erkan, U., Toktas, A., Enginoğlu, S., Akbacak, E., Thanh, D.N.H. (2022). An image encryption scheme based on chaotic logarithmic map and key generation using deep CNN. Multimedia Tools and Applications, 81: 7365-7391. https://doi.org/10.1007/s11042-021-11803-1
[10] Saini, A., Sehrawat, R. (2025). An intelligent and efficient CNN-AES framework for image block encryption with a multi-key approach. Engineering Research Express, 7: 015206. https://doi.org/10.1088/2631-8695/ada3af
[11] Zia, U., McCartney, M., Scotney, B., Martinez, J., AbuTair, M., Memon, J., Sajjad, A. (2022). Survey on image encryption techniques using chaotic maps in spatial, transform and spatiotemporal domains. International Journal of Information Security, 21(4): 917-935. https://doi.org/10.1007/s10207-022-00588-5
[12] Güvenoğlu, E. (2024). An image encryption algorithm based on multi-layered chaotic maps and its security analysis. Connection Science, 36(1): 2312108. https://doi.org/10.1080/09540091.2024.2312108
[13] Man, Z., Li, J., Di, X., Sheng, Y., Liu, Z. (2021). Double image encryption algorithm based on neural network and chaos. Chaos, Solitons & Fractals, 152: 111318. https://doi.org/10.1016/j.chaos.2021.111318
[14] Negabi, I., Ait El Asri, S., Adib, S.E., Raissouni, N. (2023). Convolutional neural network based key generation for security of data through encryption with advanced encryption standard. International Journal of Electrical and Computer Engineering, 13(3): 2589-2599. http://doi.org/10.11591/ijece.v13i3.pp2589-2599
[15] Wen, H.P., Lin, Y.T., Kang, S.H., Zhang, X.Y., Zou, K. (2024). Secure image encryption algorithm using chaos-based block permutation and weighted bit planes chain diffusion. iScience, 27: 108610. https://doi.org/10.1016/j.isci.2023.108610
[16] Saadi, Z.M., Sadiq, A.T., Akif, O.Z., Eid, M.M. (2024). Enhancing image classification using a convolutional neural network model. Journal of Soft Computing and Computer Applications, 1(2): 1010. https://doi.org/10.70403/3008-1084.1010
[17] Zhu, S., Zhu, C., Yan, H. (2023). Cryptanalyzing and improving an image encryption algorithm based on chaotic dual scrambling of pixel position and bit. Entropy, 25(3): 400. https://doi.org/10.3390/e25030400
[18] Chethan, K.S., Sumanth, K., Rajagopal, S.M. (2024). Image encryption and decryption based on 3D chaos maps and DNA encoding. In 2024 International Conference on Emerging Smart Computing and Informatics, Pune, India, pp. 1-6. https://doi.org/10.1109/esci59607.2024.10497379
[19] Yildirim, M. (2022). Optical color image encryption scheme with a novel DNA encoding algorithm based on a chaotic circuit. Chaos, Solitons & Fractals, 155: 111631. https://doi.org/10.1016/j.chaos.2021.111631
[20] Kanwal, S., Inam, S., Hajjej, F., Cheikhrouhou, O., Nawaz, Z., Waqar, A., Khan, M. (2022). A new image encryption technique based on sine map, chaotic tent map, and circulant matrices. Security and Communication Networks, 2022: 4152683. https://doi.org/10.1155/2022/4152683
[21] Guan, Z.H., Huang, F.J., Guan, W. (2005). Chaos-based image encryption algorithm. Physics Letters A, 346(1-3): 153-157. https://doi.org/10.1016/j.physleta.2005.08.006
[22] Howard, A., Sandler, M., Chen, B., Wang, W.J., et al. (2019). Searching for MobileNetV3. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), pp. 1314-1324. https://doi.org/10.1109/ICCV.2019.00140
[23] Wu, Y., Noonan, J.P., Agaian, S. (2011). NPCR and UACI randomness tests for image encryption. Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), 31-38.