Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077025

ARTICLE

A Multi-Approach Hybrid Chaos-Based Image Encryption and Steganography Algorithm Using LSB Embedding

Islam T. Almalkawi[1,*], Samer Khasawneh[1], Hamza M. Alkhatib[1], Sabya Shtaiwi[1], Rami Halloush[2] and Manel Guerrero Zapata[3]

[1]Department of Computer Engineering, Faculty of Engineering, The Hashemite University, Zarqa, Jordan
[2]College of Engineering and Technology, American University of the Middle East, Egaila, Kuwait
[3]Computer Architecture Department, Universitat Politècnica de Catalunya, Barcelona, Spain

*Corresponding Author: Islam T. Almalkawi. Email: eslam.malkawi@hu.edu.jo

Received: 01 December 2025; Accepted: 16 March 2026; Published: 13 August 2026

ABSTRACT: Current image steganography methods often struggle to balance security, payload capacity, and computational efficiency, with many spatial-domain techniques vulnerable to statistical steganalysis and complex methods incurring high overhead. To address persistent challenges in secure data communication, this paper introduces a novel hybrid chaotic-based multi-layered image security and steganography scheme to enhance resistance against detection while offering adaptable performance. The proposed scheme first integrates Fisher-Yates permutation driven by a Logistic Map PRNG, followed by stream cipher encryption using a Hénon Map-generated keystream to secure the secret image. Embedding is then performed via a unique three-pass chaotic LSB approach that utilizes chaotic-pseudo-random block selection and optimizes embedding based on bit-matching scores to enhance imperceptibility. This modular process enables a user-defined balance between computational load and image security/fidelity. Comprehensive analysis validates the scheme's effectiveness and resistance against steganalysis attacks, and performance results demonstrate that the proposed method achieves high payload capacity and superior imperceptibility, outperforming several contemporary methods. We additionally provide initial undetectability baselines using SRM and a lightweight CNN holdout detector.

KEYWORDS: Image steganography; image encryption; chaos algorithms; Hénon map; logistic map; fisher-yates; data security

1 Introduction

In this modern age of digital communication, safeguarding sensitive information remains of utmost importance. As data transmission and storage technologies advance, so do techniques to intercept and exploit these data. Encryption serves as a fundamental tool in counteracting this. Encryption transforms plaintext data into unintelligible ciphertext, rendering it indecipherable to unauthorized parties without the corresponding decryption key; with hundreds of algorithms and schemes available for use, it is only logical for it to be the dominant information hiding method. However, as technology grows, so does the computational power; what took years to break an encryption scheme two decades ago now only takes hours. Traditional encryption methods, while robust, may be the main target of malicious parties due to their limitations, particularly as quantum computing enters the scene. In response to these challenges, researchers and practitioners have turned to innovative approaches, such as chaotic encryption algorithms, to enhance the security of digital communication and storage.

Chaotic systems, which are derived from Chaos Theory [1]; characterized by sensitivity to initial conditions and a deterministic but unpredictable behavior, offer a promising avenue for encryption. Leveraging the inherent complexity and unpredictability of chaotic dynamics, chaotic encryption algorithms can generate cryptographic keys and scramble data in a manner that follows the same principles of traditional technology combined with the randomness of Chaos Theory [2]. Moreover, the integration of chaotic encryption with image steganography presents a compelling strategy to conceal sensitive information within digital images [3,4].

Steganography, the science of hidden communication, enables the embedding of secret messages or data within innocuous cover media, such as images [5]. By combining chaotic encryption with image steganography, practitioners can achieve a multilayered approach to data security, where sensitive information is not only encrypted but also concealed within visual content, further fortifying its confidentiality [6,7]. Steganography techniques are classified into spatial and frequency domain methods. Spatial methods directly alter pixel values to embed secret data. These techniques offer high embedding capacity, but are vulnerable to statistical analysis and image manipulations. Key embedding techniques include the Least Significant Bit (LSB), Pixel Value Difference (PVD), Exploiting Modification Direction (EMD), and Pixel Indicator Technique (PIT) [8,9].

Least Significant Bit (LSB) steganography is a steganography method that involves replacing the least significant bits of the cover image with the most significant (or all) bits of the secret image, thus embedding secret data without noticeably altering its overall appearance or quality [10]. By replacing these insignificant bits with the bits of the hidden message, LSB steganography allows for the seamless concealment of information within the cover media. Spatial methods are often favored for their computational efficiency and strong imperceptibility [11]. In contrast, frequency-domain techniques involve transforming image pixels to the frequency domain and embedding information by adjusting the transformed coefficients, resulting in better resistance against statistical attacks at the cost of lower embedding capacity. Frequency techniques include Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT), and Continuous Wavelet Transform (CWT) [12,13]. Hybrid methods are often used to achieve a balance capacity, invisibility, and security.

A review of existing image steganography techniques reveals significant limitations in the trade-off between imperceptibility, security, and payload capacity. Many traditional spatial-domain methods, such as naïve LSB substitution, achieve high capacity but remain highly vulnerable to statistical steganalysis (e.g., RS and Chi-square attacks) due to their predictable embedding patterns [14,15]. Conversely, while more complex approaches in the transform domain (e.g., DWT-SVD [16] or DCT [17]) or those utilizing Generative Adversarial Networks (GANs) [18] can offer improved security, they often introduce high computational overhead and may still be susceptible to targeted attacks. Furthermore, many current methods lack a robust, flexible framework, offering a static level of security and relying on simplistic key management that is insufficient for modern security demands. To address these deficiencies, this paper introduces a novel, multi-approach security algorithm that integrates chaos-based cryptography with chaos-based LSB steganography, as summarized in Fig. 1, building upon the established security enhancements provided by chaos theory [19]. Multiple chaotic map algorithms, such as the Logistic Map and Hénon Map, are utilized to ensure randomness and sensitivity to input values. The proposed method adopts the usage of metadata, produced alongside the steganographic image, and contains information used throughout the algorithm, to ensure security and correctness of data extraction on the receiver's side.

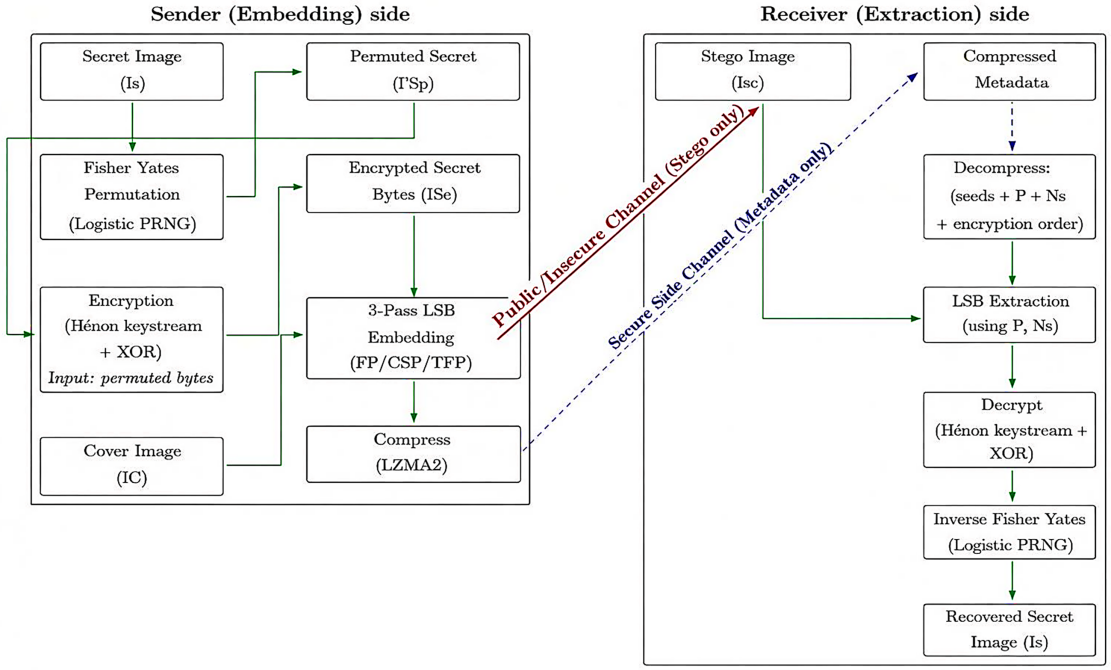


Figure 1: Complete end-to-end hybrid image steganography scheme (public stego channel + secure metadata channel).

Why combine cryptography with steganography? Steganography aims to reduce the probability of detection, but it does not guarantee confidentiality if extraction occurs. Therefore, encrypting the payload provides a second protection layer: even if an attacker detects or partially extracts embedded bits, the recovered content remains unintelligible without the keying information. This layered design is particularly relevant in realistic interception scenarios where communication channels may be monitored or logged.

Threat model and rationale for cryptographic coupling. We consider an adversary who may (i) intercept the transmitted stego-image, (ii) apply basic statistical analysis or steganalysis to suspect hidden content, and (iii) attempt to extract or tamper with the embedded payload. In such a setting, steganography alone does not guarantee confidentiality once the embedding channel is discovered. Therefore, we couple steganography with encryption so that, even under partial exposure of the embedding mechanism or successful payload extraction, the recovered data remains computationally unintelligible without the secret key(s). This layered design targets two complementary goals: (1) *concealment* of the very existence of communication (steganography), and (2) *confidentiality* of the secret content if concealment fails (cryptography). We further assume that the auxiliary metadata required for synchronization (e.g., pairing indices and chaos parameters) is delivered via a separate secure channel; the impact of metadata exposure is discussed in the limitations and future directions.

In more detail, the contribution of our paper combines:

- Chaotic-Based Encryption: A stream cipher using a Hénon Map-generated keystream to thoroughly encrypt the secret data.
- Chaotic-Based Permutation: A Fisher-Yates permutation driven by a Logistic Map to shuffle pixel data, further enhancing security.
- A Multi-Pass LSB Embedding Strategy: A unique three-stage embedding process (First Pass, Chaotic Second Pass, and Tertiary Full Pass) significantly enhances imperceptibility and resistance to statistical

attacks. The proposed algorithm can be tuned to prioritize either computational efficiency or maximum security, making it suitable for various application requirements.

- Three-pass LSB embedding (FP/CSP/TFP): The hiding stage operates on 8-bit cover-LSB blocks and proceeds in three passes. FP produces an initial chaotic pairing between secret bytes and candidate blocks. CSP then reconsiders low-score pairs and searches for alternatives with higher bit agreement, which reduces the expected number of LSB flips. Finally, TFP applies a full refinement pass for the remaining difficult cases. This staged design allows a transparent trade-off between computational cost and flip minimization (FP only, FP + CSP, or FP + CSP + TFP).
- A secure Metadata: To ensure data integrity and correct image extraction on the receiver's end, the proposed scheme generates the essential metadata required for the retrieval.

The rest of the paper is organized as follows. Section 2 reviews the background and related work. Section 3 explains the proposed multi-layered steganography scheme. Section 4 showcases the results and performance. Section 5 concludes the paper.

2 Related Work

This section outlines a selection of recent techniques introduced in the literature to implement image steganography. It concludes with a comparative analysis of some methods presented in Table 1, highlighting their main strengths and limitations in embedding secret information within images.

NSKA-LSB is a key adaptive Least Significant Bit (LSB) scheme presented in [14]. The scheme consisted of four phases of operation, namely pixel identification, secret key collection, secret embedding, and secret extraction. Pixel selection is carried out using the Knight Tour (KT) and the Hénon Map. Additionally, Run-Length Encoding (RLE) and Bernoulli's Map (BM) were used to compress and encrypt the message. In order to enhance the embedding process, direct and inverse embedding of secret bits were suggested.

Particle Swarm Optimization (PSO) is commonly used in image steganography to optimize the parameters related to embedding secret information, such as the substitution matrix. However, PSO has a premature convergence problem leading to a suboptimal solution. Jaradat et al. [20] proposed a solution for this problem by integrating the Logistic chaotic map with PSO to optimally conceal the secret data in the cover image. In this technique, the particle's speed and position were initialized using the Logistic map instead of random initialization. In addition, the cover and secret images were divided into four blocks to enhance the embedding capacity. The experimental results showed that this scheme had better Peak Signal-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) compared to existing PSO-based methods.

The authors in [21] proposed an adaptive method to select the optimal pattern to minimize error during message embedding. Using the inverted LSB substitution method, they tested different patterns on the message and container image bits, selecting the one with the minimum error rate. The results improved imperceptibility in inverted LSB image steganography. More recently, imperceptibility has also been optimized using edge-aware strategies; for instance, graded fuzzy edge detection was used to guide embedding decisions with the aim of reducing visible artifacts [22].

An image steganography approach based on LSB and Pixel Value Differencing (PVD) is proposed in [15]. The primary objective of this scheme was to prevent distortion of the carrier image, thereby allowing 100% recovery of secret data during the extraction phase. The process of pixel differencing utilized the intensities of a neighboring pixel and was applied to non-overlapping pixel blocks. Performance evaluation showed the ability of the proposed scheme to increase embedding capacity while maintaining image visual quality.

A steganographic algorithm for encrypted images is presented in [23]. The proposed algorithm aimed to tackle the problem of low security in traditional schemes by providing an encryption algorithm that was

based on a chaotic system. It ensured uniform distribution of image pixel grayscale value by integrating a double scrambling operation for the pixel position of the carrier image. The authors showed that the embedding capacity of their scheme had been improved while ensuring information security and image quality.

A fragile watermarking technique for image authentication and integrity verification based on the Logistic map is presented in [24]. This technique aimed to detect and localize tampered regions in images while balancing perceptual transparency, embedding capacity, and robustness. The watermark bits were embedded into the LSBs using a logical XOR operation, achieving a high peak signal-to-noise ratio (PSNR). The proposed pixel-wise indirect watermark embedding technique improved on previous methods by embedding watermark bits in each pixel, enhancing imperceptibility and robustness. The technique showed high accuracy in identifying tampered regions with low false positive/negative rates.

CHASE [18] is an image steganography method designed to hide multiple secret images within a single cover image. The method incorporated image scrambling, encoding, and decoding processes within the same network and used Haar wavelet transforms for better visual quality preservation. It used the Logistic map for image permutation and utilizes Generative Adversarial Networks (GANs) to enhance image fidelity. The model allowed for embedding color images in grayscale images, improving resistance to steganalysis, and ensuring good invisibility of hidden images. The training process involved a two-stage strategy, first reconstructing the secret image without a GAN and then aligning the distribution of reconstructed images with the original ones using a discriminator. The simulation results showed competitive PSNR and SSIM values. More broadly, recent AI-powered steganography research increasingly reports perceptual and distribution-level metrics (e.g., LPIPS and FID) alongside classical PSNR/SSIM, as summarized in recent survey [25].

The paper in [26] presented an image encryption scheme using the chaotic Hénon chaotic system that employed chaotic sequences for pixel rearrangement, bit replacement, and value diffusion to improve security. The evaluation result showed enhanced entropy and strong resistance to shear and noise attacks.

CIEST [27] combined cryptography and steganography with DNA encoding for enhanced digital image security. The scheme modified the cover image and used chaotic maps to improve randomness and resistance to attacks. The scheme offered high embedding capacity, low correlation, and strong security, with low Mean Square Error (MSE), high PSNR, and a large key space, making it resistant to brute-force attacks. Recent literature has also highlighted the growing interest in *compression-combined image encryption* as a practical direction to reduce transmission and storage overhead while maintaining confidentiality. Lin et al. provide a comprehensive and up-to-date review of this line of work, covering current techniques and open challenges in compression-aware encryption pipelines [28].

Table 1 lists the strengths and weaknesses of the different related research works.

Table 1: Comparison of related works in chaotic and steganographic techniques.

Ref.	Methodology	Key Techniques	Evaluation Metrics	Strengths	Weaknesses
[24]	Fragile watermarking using Logistic map	- Logistic map sensitivity to generate watermark - XOR operation on LSB	- PSNR: 51.14 - High TPR, Low FPR/FNR	- High imperceptibility - Efficient tamper detection	- Susceptible to pixel-level attacks
[26]	Chaotic-based image encryption	- Hénon chaotic system with nonlinear term - Global confusion with bit recombination	- Entropy: High - Strong resistance to attacks	- Resistant to statistical/differential attacks - Large keyspace	- High complexity due to multiple chaotic systems

(Continued)

Table 1 (continued)

Ref.	Methodology	Key Techniques	Evaluation Metrics	Strengths	Weaknesses
[27]	Combined cryptography and steganography	- DNA coding + chaotic maps - LSB embedding	- PSNR: 56.1071 - SSIM: 0.9992 - Entropy: 7.9972 - Correlation: Near zero	- High PSNR and SSIM - Strong robustness to attacks	- Computationally intensive due to multi-layer encryption
[18]	Chaotic mapping with GAN-based steganography	- Generative Adversarial Networks (GANs) - Logistic chaotic mapping	- PSNR: High - Strong steganalysis resistance	- High capacity - Strong fidelity	- Computational complexity due to GAN and chaotic mappings
[14]	New Stego Key Adaptive LSB (NSKA-LSB)	- LSB substitution - Two random functions - Chaotic and Bernoulli maps	- High PSNR, SSIM, Payload	- Good imperceptibility - Improved security through randomness	- Vulnerable if algorithm is cracked
[20]	PSO-Based Chaotic Steganography	- Particle Swarm Optimization (PSO) -Logistic chaotic map - Block-based embedding	- PSNR: 63.02 - SSIM: 0.9996	- High embedding capacity - Optimized embedding positions	- Risk of premature convergence in PSO
[23]	Reversible steganography with image interpolation and histogram shift	- Difference histogram shift - Arnold scrambling - DNA encoding - Logistic chaotic map	- PSNR: 58.92 - NPCR: 99.61% - UACI: 33.44%	- High capacity - Lossless recovery - Good quality	- High computational cost
[15]	LSB with Pixel Value Differencing (PVD)	- Adaptive LSB - PVD - Pixel Intensity Range Table	- PSNR: 35.7 - High embedding capacity	- High embedding rate - Resistance to RS analysis	- Sensitive to ML-based attacks
[29]	LSB with 1D and 2D Chaotic Maps	- LSB - Logistic and Sin-Cosine chaotic maps - Tabu Search optimization	- PSNR: 55.95 - MSE: 0.0971 - SSIM: 0.998 - High key sensitivity	- High capacity - Enhanced security	- Requires high computational power
[16]	DWT-SVD with DNA-Hyperchaotic Encryption and Fractal Cover	- DWT-SVD - DNA encryption - Fractal image cover	- PSNR: 86.15 - SSIM: 0.9943 - MSE: 0.001	- High payload - Excellent perceptual quality	- Computationally intensive (fractal generation)
[17]	Chaotic steganography with Fuzzy Inference System (FIS) and DCT	- DCT transform - Fuzzy Inference System (FIS) - Chaotic sequences	- PSNR: High - BER: Low - High capacity	- Flexible for color/grayscale - High robustness	- Requires tuning of FIS rules
[30]	Chaotic map-based steganography with DNA encoding	- DNA encoding - Tinkerbell chaotic map - LSB embedding	- PSNR: High - MSE: Low - Entropy: High - Correlation: Low	- Strong imperceptibility - Robust to attacks	- Increased computational burden

3 Proposed Method

Here, we discuss the proposed method, which employs gray-scale images for both encryption and embedding with a flexible bit-per-pixel (BPP) offering different possible bit capacity limits. The section is split into the following parts, each describing a processing stage in our method:

1. Permutation
2. Key Generation and Encryption
3. Embedding
4. Extraction and Decryption/De-permutation

The proposed method introduces a steganographic method operating on two images: a Secret Image and a Cover Image. The methodology involves two stages: (1) The Secret Image is pre-processed via permutation and encryption; (2) The resulting secured data is embedded into the Cover Image. This embedding process generates a Stego-Image, which is perceptually identical to the original Cover Image, and the associated metadata required to retrieve the hidden information. Each stage is accompanied by figures to illustrate how each stage affects the data. The image example uses Fig. 2a (Fruits) as the Secret image and Fig. 2b (Baboon) as the Cover image.

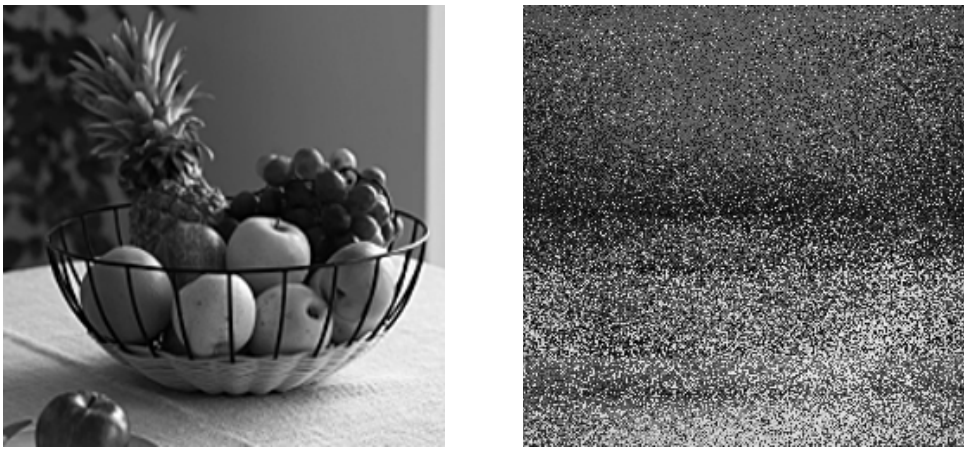
(a) Fruits **(b) Baboon**

Figure 2: Input images.

3.1 Stage 1: Permutation

In the first stage, the proposed method introduces diffusion to the secret image of size (N) pixels by shuffling all pixels into different positions. The shuffling process is performed using the Fisher-Yates algorithm [31] with the Logistic map function that operates a pseudo-random number generator (PRNG) to select the shuffling positions [32].

The shuffling process begins by positioning a pointer at the N^{th} pixel of the secret image. An iteration of the Logistic Map function is then executed, yielding a result that is subsequently processed by summing the squares of the first five decimal digits of this output. For example, if the result is 0.58162364. . ., the sum would be calculated as $5^2 + 8^2 + 1^2 + 6^2 + 2^2$. The index of a pixel designated for shuffling into the last position is determined by performing a modulus operation between this sum and the current pointer position. Following this, the two pixels are exchanged, the pointer is decremented, and the process is repeated until all pixels have been appropriately shuffled. An example of the permutation stage is shown in Fig. 3 using a fruit image. Algorithm 1 outlines the pseudo-code of the Permutation step of our proposed image steganography scheme.



(a) Before **(b) After**

Figure 3: Permutation effect.

Algorithm 1: Permutation using Fisher-Yates and Logistic Map

Require: Secret Image I^S (as a 1D array of pixels), Initial value x_0 for Logistic Map
Ensure: Permuted secret image I^{Sp}

Let N be the total number of pixels in I^S
Copy I^S to I^{Sp}
$x \leftarrow x_0$
$p \leftarrow N$
while $p > 1$ **do**
  $x \leftarrow \text{LogisticMap}(x)$ ▷ Iterate the Logistic Map
  $\text{Digits} \leftarrow \text{GetFirstFiveDecimals}(x)$ ▷ Extract first 5 decimal digits, e.g., [d1, d2, d3, d4, d5]
  $r \leftarrow \sum_{k=1}^{5}(\text{Digits}[k])^2$ ▷ Calculate sum of squares
  $s \leftarrow r \bmod p$ ▷ Calculate swap index (0 to p-1)
  $\text{swap}(I^{Sp}[s], I^{Sp}[p-1])$ ▷ Swap element at random index s with element at current end p-1
  $p \leftarrow p - 1$ ▷ Decrement the pointer
end while
return I^{Sp}

3.2 Stage 2: Key Generation and Encryption

During the encryption stage, the proposed method achieves confusion within the secret image of size N by altering pixel values rather than simply shuffling them. This is performed using a stream cipher-style encryption technique with a key stream of values ranging in [0–255] since the secret image is in grayscale.

The keystream of length N is generated by iterating the Hénon map, a well-known discrete-time chaotic system defined by $x_{t+1} = 1 - ax_t^2 + y_t$ and $y_{t+1} = bx_t$ [33]. Starting from secret initial values (x_0, y_0), the map is iterated N times. At each iteration, we convert the real-valued state into an 8-bit key byte using our deterministic digit-based quantization rule: we take the first eight decimal digits of x_t, compute the sum of their squares (analogous to the permutation stage), and apply a modulo operation with respect to 256. Recent studies have also explored strengthening chaos-based key generation by incorporating additional number-theoretic mixing mechanisms (e.g., Collatz-based constructions) to enhance key diversity in image cryptography [34].

Parameter ranges and selection. For the Logistic map $x_{t+1} = rx_t(1 - x_t)$, we select r within the chaotic regime (approximately $r \gtrsim 3.57$) and use a fixed setting $r = 3.8$ with $x_0 = 0.8$ for reproducibility. For the Hénon map, we adopt the standard chaotic parameters $(a, b) = (1.4, 0.3)$. The initial conditions (x_0, y_0) can serve as session-specific secret seeds; however, in the reported experiments we fix $(x_0, y_0) = (0.1, 0.1)$ to ensure deterministic replication. The exact settings are summarized in Table 2.

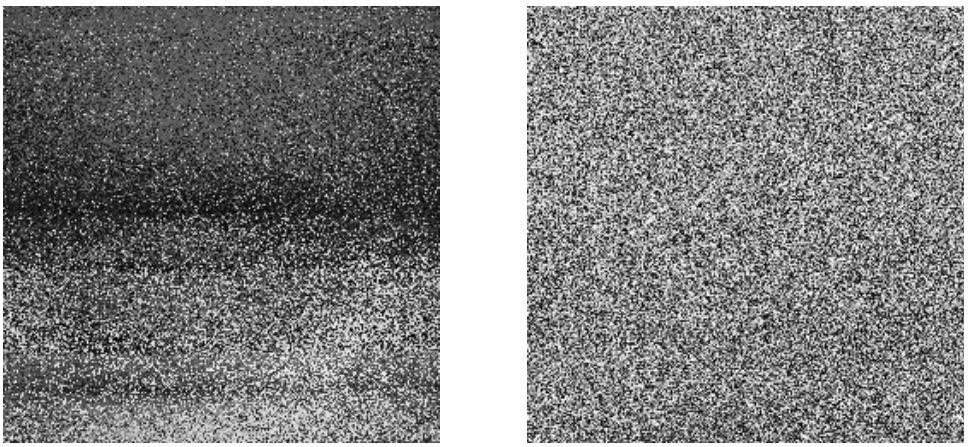
Table 2: Chaotic map parameter ranges used in our experiments.

Component	Parameter(s)	Range/Setting
Logistic map	r	3.8 (fixed; chaotic regime $r \gtrsim 3.57$)
Logistic map	x_0	0.8 (fixed; $x_0 \in (0,1)$)
Hénon map	a, b	$a = 1.4,\ b = 0.3$ (fixed; standard chaotic setting)
Hénon map	x_0, y_0	$x_0 = 0.1,\ y_0 = 0.1$ (fixed in experiments; user-provided seeds in $(0,1)$ in general)
Burn-in (optional)	T_{burn}	Not used in the embedding/encryption pipeline

In our experiments, (x_0, y_0) are fixed for reproducibility; however, they can be varied per session as secret seeds in practical deployments.

Finite-precision considerations. Chaotic maps are implemented with finite precision, which may introduce short periods or deviations from ideal real-valued dynamics. In our implementation, we use double-precision arithmetic and restrict parameters to validated chaotic ranges. An optional burn-in segment can be applied (when needed) by discarding initial iterations before sampling the keystream. A deeper treatment of quantization-induced periodicity and parameter sensitivity is left for future work.

When the key stream is generated, the proposed method encrypts the secret image using the key stream and an XOR function. This ensures that the pixel values of the secret image are effectively obscured, as shown in Fig. 4, contributing to the overall security of the encryption process. Algorithm 2 shows the generation of the secret key, and the encryption process of our proposed scheme is described in Algorithm 3.



(a) Before **(b) After**

Figure 4: Encryption effect.

Algorithm 2: Key generation

Require: Permuted secret image I^{sp}, Hénon map initial values x_0^2, y_0^2, Key stream length N
Ensure: Key stream array K
Let K be a new array of size N initialized to zero
$(x^2, y^2) \leftarrow (x_0^2, y_0^2)$ ▷ Initialize Hénon map state
for $i = 0$ to $N - 1$ **do**
  $(x^2, y^2) \leftarrow \text{HénonMap}(x^2, y^2)$ ▷ Iterate the map
  $D \leftarrow \text{ExtractDigits}(x^2, 8)$ ▷ Get first 8 decimal digits
  $r \leftarrow \sum_{j=0}^{7} (D_j)^2$ ▷ Sum the squares of the digits
  $K[i] \leftarrow r \bmod 256$
end for
return K

Algorithm 3: Encryption

Require: Permuted secret image I^{sp} (array of size N), Key stream K (array of size N)
Ensure: Encrypted secret image I^{se} (array of size N)
Let N be the size of I^{sp} (and K)
Let I^{se} be a new array of size N
for $i = 0$ to $N - 1$ **do**
  $I^{se}[i] \leftarrow I^{sp}[i] \oplus K[i]$ ▷ Perform bitwise XOR
end for
return I^{se}

3.3 Stage 3: Steganography

In this section, the different phases of our proposed image steganography scheme are explained in detail.

Block-wise embedding unit. The embedding unit is an 8-bit block (one byte): each secret pixel value is represented by 8 bits and embedded into the LSB-bitmap as one block, i.e., one secret byte per 8 cover pixels. Accordingly, each pairing address $P[j]$ indicates a cover-block index, and the starting bit position is $P[j] \times 8$.

3.3.1 Illustrative Block Calculation (Score and Flips)

To clarify block-wise embedding, consider a single secret byte represented by 8 bits embedded into the LSBs of one 8-pixel cover block. Let the secret bits be $s = 00010100$ and the cover-block LSBs be $c = 10110010$. The matching score is the number of positions where $s_i = c_i$ over 8 bits (here, score = 4/8), hence the required number of LSB modifications is 8 − score = 4. This toy example is included only for intuition; aggregated behavior across runs and configurations is reported in the ablation results in Section 4.2.1.

3.3.2 First Pass Embedding (FP)

The proposed method facilitates the selection of pixels from the encrypted secret image of size N_S to pair with pixels from the cover image. A secret image pixel is said to be "paired" when it is associated with an address that points to a sequence of pixels in the cover image. At this stage, the pixel values of the cover image of size N_C are first converted into an 8-bit binary representation. Subsequently, a bitmap is constructed by extracting the least significant bit (LSB) from each 8-bit pixel value, producing a bitmap consisting of N_C bits, capable of storing an equal number of secret data. This bit map is treated as a series of 8-bit "blocks" of size $N'_C = \lfloor N_C/8 \rfloor$, where the address of a block corresponds to the address of its first bit.

When the inputs have been prepared, the algorithm proceeds with the pairing of secret data pixels to appropriate locations within the cover image for embedding. The proposed method begins with an iteration of the Hénon Map function, multiplying the resulting output by 10^{10}, effectively shifting the decimal point 10 digits to the right. Then a modulus operation is applied to the result, using N'_C (the number of blocks) as the divisor. The outcome of this process determines the address for the first secret data pixel. This procedure is repeated with a new iteration of the Hénon Map function, utilizing the previous output for each subsequent pixel until all secret data pixels have been successfully paired with designated positions in the cover image. The above process, as explained in Algorithm 4, is henceforth referred to as "first pass embedding" or just "First Pass" (FP).

3.3.3 Chaotic Second Pass Embedding (CSP)

To produce better Steganography image quality and imperceptibility, the proposed method implements a secondary position selection and embedding procedure referred to as the "Chaotic Second Pass" (CSP). CSP begins with a bitwise comparison function, characterized by Eq. (1), which evaluates each secret data pixel against its corresponding cover bitmap block, yielding the number of matching bits between them, referred to "matching score". The function is defined as follows:

$$f(A, B) = \Sigma_{i=0}^{7} \overline{(A_i \oplus B_i)} \tag{1}$$

Algorithm 4: Bitmap creation and first pass (FP) embedding

Require: Encrypted Secret Image I^{se} (size N_s), Cover Image I^c (size N_c), Hénon map initial values (x_0^3, y_0^3)
Ensure: Steganography image I^{sc}, Metadata and extraction information P

1: Let $N'_c \leftarrow \lfloor N_c/8 \rfloor$ ▷ Number of 8-bit blocks in bitmap
2: Initialize Bitmap B (size N_c) to zero
3: Initialize Address array P (size N_s) to zero ▷ P stores target block addresses
4: $k \leftarrow 0$
5: **for** Each pixel p_i in I^c **do** ▷ Create LSB bitmap from Cover Image
6: $\quad B[k] \leftarrow \text{LSB}(\text{to}_{\text{binary}}(p_i))$

(Continued)

Algorithm 4 (continued)

$k \leftarrow k + 1$
end for
$(x^3, y^3) \leftarrow (x_0^3, y_0^3)$ ▷ Initialize Hénon state for pairing
for $j = 0$ to $N_s - 1$ **do** ▷ Generate pairing addresses for each secret pixel
    $(x^3, y^3) \leftarrow \text{HénonMap}(x^3, y^3)$
    $t \leftarrow |x^3| \times 10^{10}$ ▷ Scale the output (assuming x-value)
    $P[j] \leftarrow \lfloor t \rfloor \bmod N_c'$ ▷ Calculate target block index
end for
▷ Embed encrypted secret bits into the LSB bitmap B
for $k = 0$ to $N_s - 1$ **do**
    $E \leftarrow \text{to}_{\text{binary}}(I^{se}[k])$ ▷ Get 8 bits of secret pixel k
    $p_{start} \leftarrow P[k] \times 8$ ▷ Calculate start bit index in B using block address
    for $u = 0$ to 7 **do**
        $B[p_{start} + u] \leftarrow E[u]$ ▷ Embed secret bit u into bitmap B
    end for
end for
Let I^{sc} be a copy of I^c ▷ Prepare stego image
▷ Modify LSBs of Stego image according to the final bitmap B
for $v = 0$ to $N_c - 1$ **do**
    Set $\text{LSB}(I^{sc}[v])$ to $B[v]$ ▷ Replace LSB of cover pixel v with bitmap bit v
end for
return I^{sc}, P ▷ Return Stego Image and Pairing/Metadata

After calculating the matching scores for all secret pixels, the method identifies the worst-performing 50%, those being the pixels with the lowest matching scores, to find better positions for them. This results in a subset of size $N_S/2$ that contains the "failed" addresses. These pixels are then "unpaired", i.e., they will recalculate new positions for them among the positions paired with those 50%. The reassignment is conducted using a process similar to the First Pass but with additional scrutiny applied to the matching scores for each newly selected position. Specifically, a secret data pixel is only paired with a block if its matching score exceeds a predetermined threshold, which starts at a value of 8 and decreases by 1 if the succeeding positions are not located within a specified number of iterations. Algorithm 5 shows the pseudo-code of the CSP phase.

Algorithm 5: Chaotic Second Pass (CSP)

Require: Cover Image LSB Bitmap B (size N_c), Encrypted Secret Image I^{se} (size N_s), First Pass Pairing Addresses P (size N_s), Hénon map initial values (x_0^2, y_0^2), Initial Threshold T_{init} (e.g., 8), Iteration Limit L_{iter}
Ensure: Updated Pairing Addresses P' (size N_s)
Let $N_c' \leftarrow \lfloor N_c/8 \rfloor$ ▷ Number of 8-bit blocks in bitmap
Initialize Matching Scores array M (size N_s) to zero
$P' \leftarrow P$ ▷ Initialize Updated Addresses with First Pass results
▷ Calculate initial matching scores
for $i = 0$ to $N_s - 1$ **do**

(Continued)

Algorithm 5 (continued)

$p \leftarrow P[i]$ ▷ Get block address from First Pass
$m \leftarrow$ Extract 8 bits from B starting at $p \times 8$ ▷ Get cover block byte
$M[i] \leftarrow \text{MatchScore}(I^{se}[i], m)$ ▷ Calculate score
end for
$F \leftarrow \text{SelectLowestHalfIndices}(M)$ ▷ Get indices of the lowest 50% scores
Let $N_{low} \leftarrow \lfloor N_s/2 \rfloor$ ▷ Number of low-scoring pixels
Let $P_{candidate}$ be the list of addresses from $P[i]$ where $i \in F$ ▷ Create candidate pool
Let $PoolSize \leftarrow N_{low}$
$(x^2, y^2) \leftarrow (x_0^2, y_0^2)$ ▷ Initialize Hénon state for re-pairing
$threshold \leftarrow T_{init}$
$iteration\_count \leftarrow 0$
for each index j in F **do** ▷ Loop through low-scoring pixel indices
  $position\_found \leftarrow$ FALSE
  $current\_secret\_pixel \leftarrow I^{se}[j]$
  $threshold \leftarrow T_{init}$ ▷ Reset threshold for each pixel
  $iteration\_count \leftarrow 0$ ▷ Reset iteration count for each pixel
  while $position\_found$ = FALSE **and** $threshold \geq 0$ **do**
    $(x^2, y^2) \leftarrow$ Hé nonMap(x^2, y^2)
    $t \leftarrow |x^2| \times 10^{10}$ ▷ Scale Hénon output
    $candidate\_pool\_index \leftarrow \lfloor t \rfloor \bmod PoolSize$ ▷ Get index into candidate pool
    $candidate\_address \leftarrow P_{candidate}[candidate\_pool\_index]$ ▷ Pick address from pool
    $candidate\_block \leftarrow$ Extract 8 bits from B starting at $candidate\_address \times 8$
    $current\_score \leftarrow \text{MatchScore}(current\_secret\_pixel, candidate\_block)$
    if $current\_score \geq threshold$ **then**
      $P'[j] \leftarrow candidate\_address$ ▷ Update address
      $position\_found \leftarrow$ TRUE
    else
      $iteration\_count \leftarrow iteration\_count + 1$
      if $iteration\_count \geq L_{iter}$ **then**
        $threshold \leftarrow threshold - 1$ ▷ Decrease threshold
        $iteration\_count \leftarrow 0$ ▷ Reset count for new threshold
      end if
    end if
  end while
end for
return P' ▷ Return the updated pairing addresses

3.3.4 The Tertiary Full Pass (TFP)

To further enhance the output image quality, the proposed method includes an optional tertiary procedure named the Tertiary Full Pass (TFP). The TFP possesses a structure similar to the CSP but differs in two key aspects, as shown in Algorithm 6: (1) it employs a linear search mechanism and (2) it executes a full fixing procedure for any secret pixel that yields a matching score below 8. The effect of this optional stage on embedding quality and computational complexity is evaluated in the Experimental Results Section. Fig. 5

shows the flow diagram of our proposed image steganography method. Fig. 6 shows the cover image after hiding the secret data using the proposed (FP, CSP, and TFP) embedding phases.

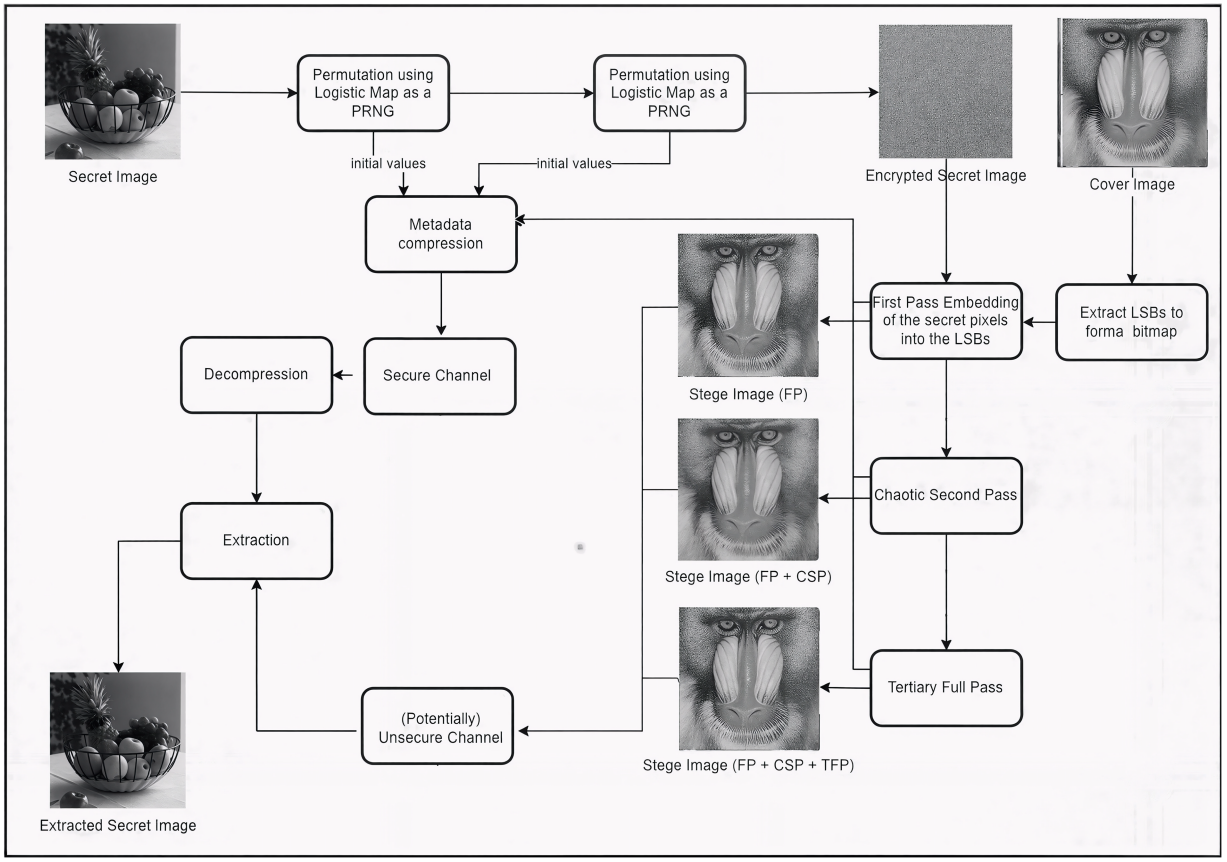
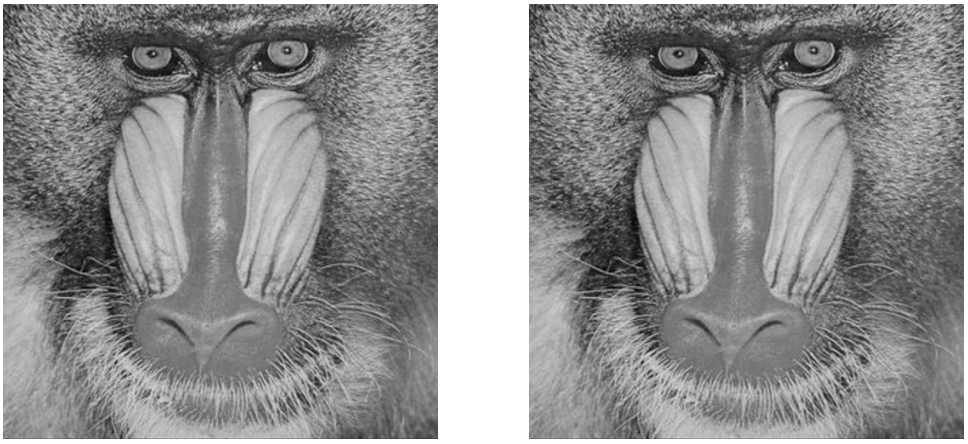


Figure 5: Flowchart of our proposed method.



(a) Before **(b) After**

Figure 6: Embedding effect (FP + CSP + TFP).

Algorithm 6: Tertiary Full Pass (TFP)

Require: Cover Image LSB Bitmap B (size N_c), Encrypted Secret Image I^{se} (size N_s), Current Pairing Addresses $P_{current}$ (size N_s, from FP or CSP)
Ensure: Final Pairing Addresses P_{final} (size N_s), Updated Stego Image I^{sc}

Let $N'_c \leftarrow \lfloor N_c/8 \rfloor$ ▷ Number of 8-bit blocks in bitmap
Initialize Matching Scores array M (size N_s) to zero
$P_{final} \leftarrow P_{current}$ ▷ Initialize Final Addresses
▷ Calculate initial matching scores based on current pairings
for $i = 0$ to $N_s - 1$ **do**
  $p_{addr} \leftarrow P_{current}[i]$ ▷ Get current block address
  $m \leftarrow$ Extract 8 bits from B starting at $p_{addr} \times 8$ ▷ Get cover block byte
  $M[i] \leftarrow \text{MatchScore}(I^{se}[i], m)$ ▷ Calculate score
end for
Let $IndicesToFix$ be the list of indices i where $M[i] < 8$
▷ Attempt linear search for better positions for low-scoring pixels
for each index j in $IndicesToFix$ **do**
  $current\_secret\_pixel \leftarrow I^{se}[j]$
  $position\_found \leftarrow$ FALSE
  for $candidate\_address = 0$ to $N'_c - 1$ **do** ▷ Linear search through all blocks
    $candidate\_block \leftarrow$ Extract 8 bits from B starting at $candidate\_address \times 8$
    $current\_score \leftarrow \text{MatchScore}(current\_secret\_pixel, candidate\_block)$
    if $current\_score \geq 8$ **then**
      $P_{final}[j] \leftarrow candidate\_address$ ▷ Update address
      $position\_found \leftarrow$ TRUE
      break ▷ Exit inner loop once a good position is found
    end if
  end for
  if $position\_found ==$ FALSE **then**
    Keep the current pairing address $P_{final}[j]$ (no update) and proceed to the next index
  end if
end for
▷ Re-embed all secret bits into Bitmap B using final addresses P_final
Initialize Bitmap B_{final} (size N_c) based on original Cover Image I^c LSBs
for $w = 0$ to $N_s - 1$ **do**
  $E \leftarrow \text{to}_{\text{binary}}(I^{se}[w])$ ▷ Get 8 bits of secret pixel w
  $p_{start} \leftarrow P_{final}[w] \times 8$ ▷ Calculate start bit index using final address
  for $u = 0$ to 7 **do**
    $B_{final}[p_{start} + u] \leftarrow E[u]$ ▷ Embed secret bit u into final bitmap
  end for
end for
Let I^{sc} be a copy of the original Cover Image I^c ▷ Prepare final stego image
▷ Modify LSBs of Stego image according to the final bitmap B_final
for $v = 0$ to $N_c - 1$ **do**

(Continued)

Algorithm 6 (continued)

40: Set LSB($I^{sc}[v]$) to $B_{final}[v]$ ▷ Replace LSB
41: **end for**
42: **return** I^{sc}, P_{final} ▷ Return Final Stego Image and Final Pairings

After the proposed method generates the steganographic image, the method will generate a file alongside the stego-image that contains important information employed in the algorithm; the file is referred to as "metadata" [35]. Metadata for the proposed method consists of the following:

1. Initial values of the chaotic algorithms used in the permutation, encryption, and embedding stage.
2. Pixel pairing pattern produced in the embedding stage.

Similar to symmetric keys used in encryption algorithms such as AES (Advanced Encryption Standard) [36], which are shared through advanced key exchange protocols [37], the metadata in the proposed method is also sent through a different, preferably secure, channel than the one used to send the stego-image. To facilitate efficient distribution, Similar to symmetric keys used in encryption algorithms such as AES (Advanced Encryption Standard) [36], which are shared through advanced key exchange protocols [37], the metadata in the proposed method is also sent through a different, preferably secure, channel than the one used to send the stego-image. To facilitate efficient distribution, we compress *only the metadata file* using the Lempel-Ziv-Markov (LZMA2) algorithm [38]. This step is applied exclusively to the side information (e.g., chaotic parameters and pairing pattern), and is *not* applied to the cover image, the stego-image, or the encrypted payload. LZMA2 can provide substantial size reduction depending on the metadata structure, thereby reducing the transmission overhead of the auxiliary file. During the extraction phase, the compressed metadata is decompressed to recover the original information required for the Extraction, Decryption, and De-permutation of the stego-image. We treat secure transmission of metadata (e.g., via an authenticated and encrypted channel) as a practical system assumption; a full treatment of key exchange and compression internals is outside the scope of this paper and will not be discussed in further detail.

3.3.5 Quantitative Metadata Overhead and Exposure Implications

Our stego construction produces a stego image accompanied by auxiliary information required for correct extraction. In our implementation, the embedding stage returns a metadata dictionary that includes: the secret dimensions (w_s, h_s), the payload size in bytes N_s (`payload_bytes`), and an addressing (pairing) list P (stored in the metadata field `adds`, i.e., an address list with one entry per embedded secret byte) that specifies where each secret byte is embedded. Since we store one address per secret byte, the list length satisfies:

$$|P| = N_s. \tag{2}$$

Metadata size as a function of payload

In our reference implementation, the addressing list P is serialized as a 32-bit integer array (each address stored as `uint32`) and concatenated with a small constant-size header. Therefore, the raw metadata size scales linearly with payload:

$$\text{Meta}_{\text{raw}}(N_s) \approx 4N_s + C, \tag{3}$$

where $4N_s$ accounts for the `uint32` address list and C is a small constant header overhead. We further report compressed metadata size using lossless `zlib` compression applied to the serialized metadata (see Table 3

and Algorithm 7). *Note that Table 3 reports averages over multiple payload sizes across runs; hence the displayed raw size corresponds to* $4\overline{N_s} + C$ *rather than a single maximum-payload instance.*

Table 3: Metadata size before/after compression (bytes). Compression is applied to the serialized extraction metadata (address list and compact header fields). Values are reported as averages over runs spanning multiple cover/secret sizes. (Optimal values in bold).

Mode	Enc	Raw (B)↓	Compressed (B)↓	Saving (%)↑	Ratio (Comp/Raw)↓
FP	0	**70,542**	46,343	34.30	**0.657**
FP_CSP	0	70,546	**46,337**	**34.32**	**0.657**
FP_CSP_TFP	0	70,550	46,348	34.30	**0.657**
FP	1	**70,542**	46,343	34.30	**0.657**
FP_CSP	1	70,546	46,344	34.31	**0.657**
FP_CSP_TFP	1	70,550	46,350	34.30	**0.657**

Does metadata growth become combinatorial?

The *size* of transmitted metadata is linear because we serialize and transmit an explicit list of N_s addresses. The *combinatorial* term applies to the theoretical number of possible address patterns, but it does not change the transmitted file length under this explicit-list representation.

Security implications under metadata exposure

To avoid ambiguity, we explicitly separate:

- **Extraction side-information:** (w_s, h_s), N_s, and the addressing list P.
- **Confidentiality-critical key material:** the secret seeds/keys used to generate the permutation and keystream for payload encryption.

If an attacker obtains only the extraction side-information (e.g., P and (w_s, h_s)), they may extract the *encrypted* payload (ciphertext) from the stego image, but cannot reconstruct the plaintext secret without the encryption key material. Conversely, if the encryption key material is compromised, confidentiality is lost. Accordingly, our threat model assumes that confidentiality-critical key material is protected by an authenticated and encrypted channel.

Implementation-grounded summary

In short: (i) metadata size scales as $\approx 4N_s + C$ due to the `uint32` address list, (ii) growth is linear (not combinatorial) under our explicit-list representation, and (iii) exposure of extraction side-information yields at most ciphertext recovery, while plaintext confidentiality depends on keeping the encryption key material secret.

Algorithm 7: Metadata serialization and size scaling (Implementation-Aligned)

Require: Payload size N_s (bytes), address list P (length N_s), header fields
Ensure: $\mathcal{M}_{raw}$, $\mathcal{M}_{cmp}$, and sizes $|\mathcal{M}_{raw}|$, $|\mathcal{M}_{cmp}|$
Serialize header fields into a byte string H ▷ $|H|$ is constant-size
$A \leftarrow \texttt{uint32}(P)$ ▷ $|A| = 4N_s$ bytes
$\mathcal{M}_{raw} \leftarrow H \,\|\, A$
$\mathcal{M}_{cmp} \leftarrow \texttt{zlib}(\mathcal{M}_{raw})$
return $\mathcal{M}_{raw}, \mathcal{M}_{cmp}, |\mathcal{M}_{raw}|, |\mathcal{M}_{cmp}|$

3.4 Stage 4: Data Extraction

To extract the secret image, the initial values and pairing pattern are first extracted from the decompressed metadata; this will include:

1. Initial values for the Logistic Map for de-permutation x_0^1.
2. Initial values for the Hénon Maps in the decryption and extraction $(x_0^2, y_0^2), (x_0^3, y_0^3)$.
3. Size of the expected secret image N_S.
4. Pairing pattern for extraction P.

Threat model for metadata exposure. Our extraction procedure relies on auxiliary metadata (e.g., chaotic seeds and the pairing pattern). If this metadata is disclosed to an adversary, recoverability may be compromised and confidentiality may degrade depending on the configuration. Accordingly, we assume that metadata is transmitted over an authenticated and encrypted channel (e.g., a secure out-of-band session). A complete key-management design is beyond the scope of this work; we state this assumption to delimit the threat model.

Metadata size reduction. Metadata compression is applied *only* to the side information (chaotic seeds and the address list), not to the cover image nor the payload. Table 3 reports raw vs. compressed metadata sizes (bytes) under each configuration.

The sequence of secret bits is then extracted from the stego-image I^{sc} of size N_C using the corresponding values and pattern to obtain the secret image. Algorithms 8 and 9 show the pseudo-code of the data extraction process and de-permutation. The decryption pseudo-code is identical to Algorithms 2 and 3.

Algorithm 8: Extraction of secret data

Require: Steganography image I^{sc} (size N_c), Metadata/Pairing Addresses P (size N_s)
Ensure: Extracted secret image I^s (size N_s)
Let N_s be the size required for I^s (length of P)
Let N_c be the size of I^{sc}
Initialize extracted image array I^s (size N_s) to zero
Initialize Bitmap B (size N_c) to zero
$k \leftarrow 0$
for Each pixel p_i in I^{sc} **do** ▷ Reconstruct LSB bitmap from Stego Image
    $B[k] \leftarrow \text{LSB}(\text{to}_{\text{binary}}(p_i))$
    $k \leftarrow k + 1$
end for
▷ Extract secret pixel bits from Bitmap B using pairing addresses P
for $j = 0$ to $N_s - 1$ **do**
    $p_{start} \leftarrow P[j] \times 8$ ▷ Get start bit index from block address P[j]
    Let $extracted\_bits$ be a temporary array of 8 bits
    for $k = 0$ to 7 **do**
        $extracted\_bits[k] \leftarrow B[p_{start} + k]$ ▷ Extract the k-th bit for pixel j
    end for
    $I^s[j] \leftarrow \text{AssembleBitsToByte}(extracted\_bits)$ ▷ Assemble bits into pixel value
end for
return I^s ▷ Return the extracted (still encrypted) secret image

Algorithm 9: Correct De-permutation (Inverse Fisher-Yates for Algorithm 1)

Require: Permuted Image I^{sp} (size N), Initial value x_0 for Logistic Map
Ensure: Original Secret Image I^S
Let N be the total number of pixels in I^{sp}
Copy I^{sp} to I^S ▷ Regenerate the sequence of Logistic Map values used during permutation
Let X be an array of size N
$x_{temp} \leftarrow x_0$
for $p = N$ down to 2 **do**
  $x_{temp} \leftarrow \text{LogisticMap}(x_{temp})$
  $X[p-1] \leftarrow x_{temp}$
end for ▷ Perform swaps in reverse order
for $p = 2$ to N **do**
  $x \leftarrow X[p-1]$
  Digits $\leftarrow$ GetFirstFiveDecimals(x)
  $r \leftarrow \sum_{k=1}^{5}(\text{Digits}[k])^2$
  $s \leftarrow r \bmod p$ ▷ Calculate swap index used at step p during permutation
  swap$(I^S[s], I^S[p-1])$ ▷ Perform the same swap again
end for
return I^S

4 Experimental Results

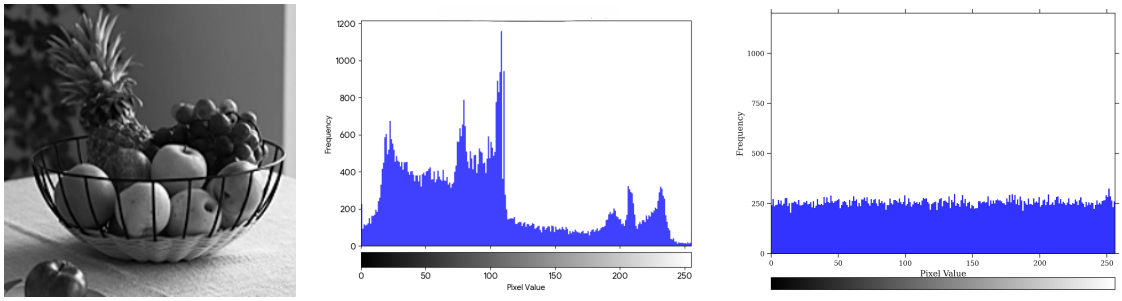
The detailed performance comparison between the proposed approach and some existing schemes is presented in this section. We analyzed the impact of different embedding procedures on the performance of the proposed technique. The embedding and extracting simulations were conducted using the USC-SIPI [39] standard image database. The image dataset contains uncompressed colored images with various textures and resolutions (512 × 512 and 1024 × 1024). For experiments, color images are transformed into gray-scale as shown in Fig. 7 (e.g., Boat, Pepper, Baboon). Unless stated otherwise, all reported results are averaged over multiple covers and multiple secret images to reduce bias towards a single image characteristic. Specifically, we use two secret images (Fruits and Baboon) and two cover images (Boat and Peppers), across payload sizes up to 180 × 180.

(a) Baboon (b) Peppers (c) Boat

Figure 7: Examples of cover images.

4.1 Histogram Result Analysis

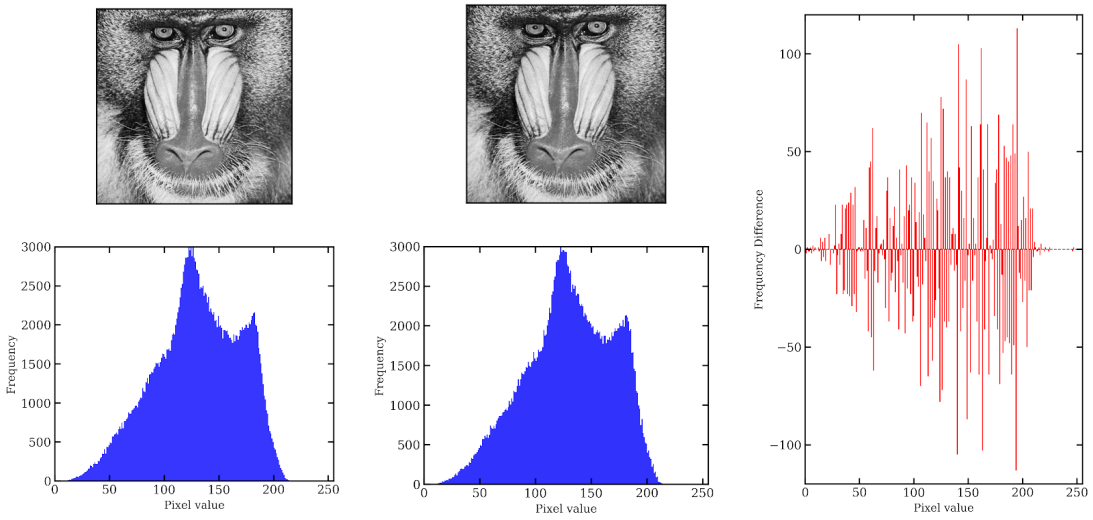
In this subsection, the histogram figures of the images are presented. The histogram distribution shows the frequency of pixel intensities in the image, where 0 represents black and 255 represents white. An attacker (frequency attacks) can use this statistical information to find the content of the image. Fig. 8 shows the histogram analysis of the secret image, Fruits, before and after the "Permutation" and "Encryption" stages. The pre-steganography process aims to flatten the image histogram into a uniform distribution.



(a) Secret Image (b) Original (c) Encrypted

Figure 8: Histograms of the secret image before and after encryption.

In addition, the primary goal of our image steganography is imperceptibility. Therefore, the histogram of the "Stego Image" should look statistically indistinguishable from the "Cover Image" and prevent histogram attacks. Fig. 9 compares (a) the histogram of the "Baboon" cover image with (b) the histogram of the "Baboon" stego image. As evident from the histogram data shown, the proposed steganography scheme not only maintains perceptual similarity between the cover and stego images but also has successfully hidden the secret image without destroying the global statistical properties of the cover image. Fig. 9c shows the difference histogram (Stego-Cover) with minor fluctuations, centered around zero, which confirms negligible statistical distortion and proves good resistance to statistical attacks (like Chi-square).



(a) Cover image and histogram **(b)** Stego image and histogram **(c)** Histogram difference

Figure 9: Histogram analysis of the cover and stego images.

4.2 Simulation and Image Quality Metric Result Analysis

We evaluate below the simulation results of the proposed algorithm and compare it with other innovative chaotic steganography methods. An important performance metric to measure is the Bit Capacity, the amount of bits embedded into the cover image. This is dependent on both the number of bits embedded per pixel (BPP) [40] and the resolution of the secret image. The actual quality of the results can be measured using several metrics, such as PSNR (Peak Signal-Noise Ratio) and SSIM (Structural Similarity Index), and other metrics like Pixel Change Rate (NPCR) and Averaged Changing Intensity (UACI).

The Peak Signal-to-Noise Ratio (PSNR) compares the quality of the images by measuring the average difference between their pixel values and those of the originals [41]. Characterized by Eq. (4), PSNR leverages another metric called Mean Square Error (MSE), which measures the average squared difference between the original and processed images [42]. MSE and PSNR have an inverse relation; a lower MSE, characterized by Eq. (5), results in a higher PSNR, and that in turn means less distortion and better stego-image quality.

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{4}$$

$$\text{MSE} = \frac{1}{W \times H}\sum_{i=1}^{H}\sum_{j=1}^{W}(I(i,j) - J(i,j))^2 \tag{5}$$

where $I(i,j)$ and $J(i,j)$ are the pixel values of the original and processed images, respectively, and W, H are the width and height of the images (which should be the same), and MAX_I is the maximum possible pixel intensity (e.g., 255 for 8-bit images).

The Structural Similarity Index Measure (SSIM) measures the similarity between two images by evaluating things like contrast, structural information, and luminance [43]. With values ranging between 0 and 1, where 1 means near-identical images, SSIM is characterized in Eq. (6) as follows:

$$\text{SSIM}(I,J) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{6}$$

where μ is the mean pixel value, σ is the standard deviation, and σ_{IJ} is the covariance of the two images, and C_1 and C_2 are constants to stabilize the division.

NPCR and UACI, defined in Eqs. (7) and (8), are related image quality metrics that measure the change in pixel values after a small change in the key or steganography, the addition of secret pixels. Where NPCR measures it as a percentage, with lower percentages suggesting minimal change in the image, and UACI measures it as an absolute difference, where lower values mean minimal distortion [44].

$$\text{NPCR} = \frac{100}{W \times H}\sum_{i=1}^{H}\sum_{j=1}^{W}\delta(I(i,j), J(i,j)) \tag{7}$$

$$\text{UACI} = \frac{100}{W \times H \times 255}\sum_{i=1}^{H}\sum_{j=1}^{W}|I(i,j) - J(i,j)| \tag{8}$$

where $\delta(I(i,j), J(i,j)) = 1$ if the pixel values are different and 0 if they are the same.

The performance and quality metrics of the proposed method are first evaluated across several factors discussed above.

The results in Table 4 were produced by running the proposed methods with all embedding procedures (FP, CSP, TFP). The cover images had the resolution of (512 × 512) while the secret images were used with three different resolutions: (180 × 180), (220 × 220), (256 × 256), each producing bit capacities of 259,200, 387,200, and 524,288, respectively.

Table 4: Quality metrics for various cover images using the proposed method. (Optimal values in bold).

Cover Image	Bit Capacity	BPP	PSNR↑	MSE↓	SSIM↑	NPCR↑	UACI↑	Entropy↑
Baboon	387,200	2	74.734	0.0021	0.9999992	**0.0858**	**0.00051**	7.26650
	524,288	2	74.992	0.0020	0.9999993	0.0835	0.00048	7.26649
Peppers	259,200	1	**78.872**	**0.00084**	**0.9999998**	0.0843	0.00033	**7.59360**
	387,200	2	74.727	0.0021	0.9999996	**0.0873**	**0.00051**	**7.59360**
	524,288	2	74.928	0.0020	0.9999996	0.0854	0.00049	7.59359
Boat	259,200	1	**78.532**	**0.00091**	**0.9999997**	**0.0911**	0.00035	7.19141
	387,200	2	74.6298	0.0022	0.9999994	0.0862	**0.00050**	**7.19143**
	524,288	2	74.944	0.0020	0.9999995	0.0846	0.00049	7.19141

The performance of the proposed method using all three embedding procedures (FP + CSP + TFP) and a secret image of size (220 × 220) is compared with other chaotic-based methods: Hussein et al. in [15], Rustad et al. in [21], Jaradat et al. in [20], and Abdulwahed in [14]. Tables 5 and 6, which compare the PSNR and SSIM values, respectively, indicate that the proposed method yields better quality results. For example, the result for the Baboon image shows that the proposed method results in a PSNR and SSIM of 74.734 and 0.9999992, respectively, while Abdulwahed et al managed to result in PSNR and SSIM of 72.35 and 0.990, respectively.

Table 5: PSNR comparison. (Optimal values in bold).

Cover Image	Proposed Method	Abdulwahed [14]	Jaradat et al. [20]	Rustad et al. [21]	Hussein et al. [15]
Baboon	**74.734**	72.35	63.253	57.01	34.10
Peppers	**74.727**	72.44	63.16	57.41	36.07

Table 6: SSIM comparison. (Optimal values in bold).

Cover Image	Proposed Method	Abdulwahed [14]	Rustad et al. [21]	Jaradat et al. [20]
Baboon	**0.9999992**	0.9900	0.999645	0.9998
Peppers	**0.9999996**	0.9800	0.999275	N/A

4.2.1 Ablation Study of the Multi-Pass Embedding Stages (FP/CSP/TFP)

To quantify the contribution of each embedding stage, we evaluate three configurations: FP, FP + CSP, and FP + CSP + TFP. We report results (i) without encryption to isolate embedding behavior, and (ii) with encryption enabled to reflect the full pipeline. For fair reference, we also include a naive sequential LSB baseline (no chaotic addressing and no refinement).

Overall, CSP and TFP reduce the embedded-region bit-flip rate, which correspondingly lowers MSE and increases PSNR/SSIM, at the cost of additional runtime. To reduce any potential cherry-picking, we also report worst-case outcomes under the same protocol (Section 4.2.2).

Additionally, we report learned perceptual similarity (LPIPS) and distribution-level quality (FID). As shown in Tables 7–9, CSP/TFP consistently reduce perceptual distortion (LPIPS) and improve set-level fidelity (FID), consistent with the reduced embedded-region bit-flip rates.

Table 7: Ablation results without encryption (mean over all covers/secrets/sizes). (Optimal values in bold).

Config	PSNR↑ (dB)	MSE↓	SSIM↑	LPIPS↓	Bit-Flips↓ (%)	Time↓ (s)	Meta(comp)↓ (B)
Naive LSB (sequential)	53.833	0.269035	0.999947	–	50.039	0.248	**18,970**
FP	53.831	0.269137	0.999947	0.000429	50.062	**0.207**	46,343
FP + CSP	57.286	0.121487	0.999976	0.000190	21.955	20.817	46,337
FP + CSP + TFP	**60.836**	**0.053637**	**0.999989**	**0.000085**	**5.678**	60.904	46,348

Note: PSNR is computed from the averaged MSE to avoid ∞ averages when some runs yield MSE = 0 (near-lossless cases). Bit-flips are measured over the embedded region (8 pixels per embedded byte). Meta(comp) reports the compressed size of side-information only (seeds + address list).

Table 8: Ablation results with encryption enabled (mean over all covers/secrets/sizes). (Optimal values in bold).

Config	PSNR↑ (dB)	MSE↓	SSIM↑	LPIPS↓	Bit-flips↓ (%)	Time↓ (s)	Meta(comp)↓ (B)
Naive LSB (sequential)	53.834	0.268975	0.999947	–	50.014	0.247	**18,968**
FP	53.837	0.268771	0.999947	0.000443	49.975	**0.212**	46,343
FP + CSP	57.609	0.112761	0.999977	0.000175	19.698	18.199	46,344
FP + CSP + TFP	**62.109**	**0.040016**	**0.999992**	**0.000061**	**4.055**	55.923	46,350

Note: PSNR is computed from the averaged MSE (stable under near-lossless cases). Bit-flips are measured over the embedded region. Meta(comp) reports compressed side-information only.

Table 9: Approximate FID between cover and stego image sets (lower is better). For grayscale images, the single channel is replicated to 3 channels for feature extraction. (Optimal values in bold).

Mode	Enc	*n* pairs	FID↓
FP	0	18	0.257260
FP_CSP	0	18	0.136559
FP_CSP_TFP	0	18	**0.079522**
FP	1	18	0.272970
FP_CSP	1	18	0.134044
FP_CSP_TFP	1	18	**0.055530**

4.2.2 Worst-Case Analysis (Anti-Cherry-Picking)

In addition to mean results, we report worst-case outcomes across all executed runs (covers × secrets × sizes). Specifically, we highlight the minimum PSNR (equivalently, the maximum MSE) and the maximum embedded-region bit-flip rate observed under each configuration.

4.2.3 Why Very High PSNR can Occur in LSB-Based Embedding

Our embedding modifies only the least significant bit (LSB) of selected pixels. Therefore, each modified pixel changes by at most 1 gray level. If a fraction p of pixels is modified, the expected mean-squared error scales with both p and the squared change magnitude, yielding very small MSE in near-lossless settings.

Consequently, when CSP/TFP substantially reduces the embedded-region bit-flip rate, MSE becomes extremely small and PSNR can exceed 70 dB, consistent with the reported ablation results.

4.2.4 Bit-Flip Distribution (Anti-Cherry-Picking)

In addition to mean values, we report the distribution of the embedded-region bit-flip rate across all executed runs. Fig. 10 shows that the refinement stages (CSP/TFP) shift the distribution towards lower flip rates compared to FP, confirming that multi-pass score optimization reduces the expected number of LSB modifications. Table 10 provides a quantitative summary of this distribution (min, quartiles, median, and max) to complement the visual analysis and avoid cherry-picking based on averages alone.

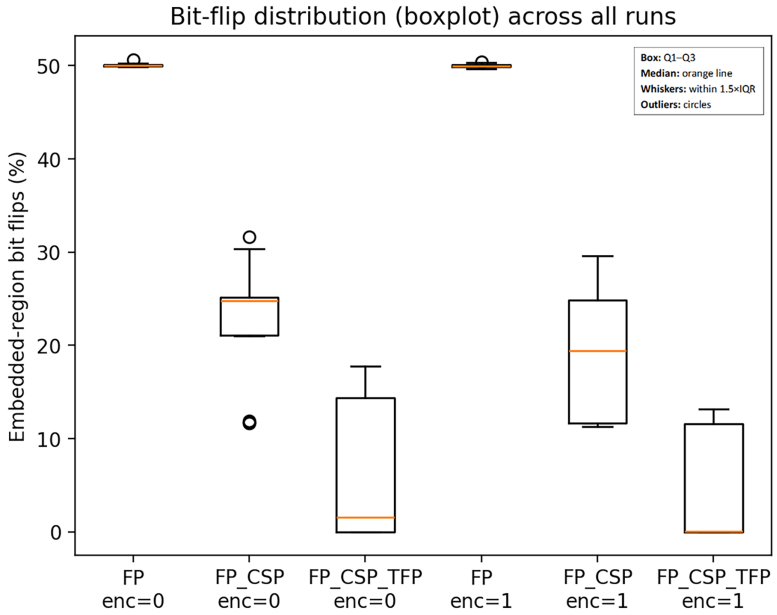


Figure 10: Embedded-region bit-flip rate distribution (boxplot) across all runs for FP variants, with/without encryption. The box spans Q1–Q3, the orange line is the median, whiskers show the non-outlier range (1.5 × IQR), and circles denote outliers.

Table 10: Embedded-region bit-flip distribution summary (quantiles, %, lower is better) across all executed runs. (Optimal values in bold).

Mode	Enc	Min↓	Q1↓	Median↓	Q3↓	Max↓
FP	0	49.853	49.947	50.029	50.092	50.665
FP_CSP	0	11.643	21.042	24.779	25.115	31.642
FP_CSP_TFP	0	**0.000**	**0.003**	**1.525**	**14.396**	**17.725**
FP	1	49.652	49.855	49.965	50.071	50.430
FP_CSP	1	11.292	11.625	19.393	24.835	29.568
FP_CSP_TFP	1	**0.000**	**0.001**	**0.048**	**11.603**	**13.157**

4.2.5 Payload Scaling and Runtime Growth

Table 11 provides the full payload-scaling summary across all settings.

Table 11: Payload scaling summary (mean over all covers/secrets for each resized payload) under encryption disabled/enabled. (Optimal values in bold).

Size	Enc	Config	Payload (B)	PSNR↑ (dB)	MSE↓	Bit-Flips↓ (%)	Time↓ (s)	Meta(comp)↓ (B)
64	0	FP	4096	60.16	0.062663	50.131	**0.18**	**10,878**
		FP + CSP	4096	63.59	0.028476	22.781	1.40	10,881
		FP + CSP+ TFP	4096	**107.09**	**0.000001**	**0.001**	14.95	10,887
64	1	FP	4096	60.17	0.062488	49.990	**0.20**	**10878**
		FP + CSP	4096	64.57	0.022720	18.176	1.58	10,884
		FP + CSP + TFP	4096	**110.10**	**0.000001**	**0.001**	16.91	10,885
128	0	FP	16,384	54.15	0.250200	50.040	**0.21**	**43,134**
		FP + CSP	16,384	58.49	0.092102	18.420	11.58	43,136
		FP + CSP + TFP	16,384	**69.28**	**0.007681**	**1.536**	65.83	43,137
128	1	FP	16,384	54.15	0.249898	49.980	**0.20**	43,134
		FP + CSP	16,384	58.54	0.091056	18.211	6.96	**43,132**
		FP + CSP+ TFP	16,384	**84.26**	**0.000244**	**0.049**	52.49	43,141
180	0	FP	32,400	51.19	0.494546	50.016	**0.24**	85,016
		FP + CSP	32,400	54.26	0.243882	24.665	49.48	**84,995**
		FP + CSP+ TFP	32,400	**56.28**	**0.153229**	**15.497**	101.93	85,021
180	1	FP	32,400	51.19	0.493928	49.954	**0.24**	**85,016**
		FP + CSP	32,400	54.62	0.224505	22.706	46.05	85,016
		FP + CSP + TFP	32,400	**57.35**	**0.119802**	**12.116**	98.36	85,025

Note: Enc = 0/1 denotes encryption disabled/enabled. PSNR is computed from the averaged MSE (stable under near-lossless cases).

4.2.6 Robustness under Common Image Operations

We additionally evaluate robustness under common channel operations, including JPEG compression (Q95/Q85/Q75), Gaussian blur (radius 0.5/1.0), and resizing (0.75× and 0.5× downscale followed by upscaling). We report the bit error rate (BER) and exact recovery after extraction. This experiment is included to delimit the threat model: as expected for LSB-based schemes, lossy operations disrupt embedded LSBs and can prevent exact recovery. Detailed results are reported in Table 12.

Table 12: Robustness evaluation under common operations at the largest payload (180 × 180, payload = 32400 B). Values are mean BER across covers and secret images (lower is better). Enc = 0/1 denotes encryption disabled/enabled. Exact recovery was 0 for all tested operations, consistent with the known non-robustness of LSB embedding under lossy/resampling transforms. (Optimal values in bold).

Attack	Level	FP (Enc0)	FP + CSP (Enc0)	FP + CSP + TFP (Enc0)	FP (Enc1)	FP + CSP (Enc1)	FP + CSP + TFP (Enc1)
BLUR	R0.5	0.289	0.286	**0.285**	0.289	0.287	**0.285**
BLUR	R1.0	0.498	0.498	**0.497**	0.498	0.498	**0.497**
JPEG	Q75	0.500	0.500	**0.499**	**0.500**	0.501	**0.500**
JPEG	Q85	0.501	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**
JPEG	Q95	**0.499**	0.500	0.501	0.501	**0.500**	**0.500**
RESIZE	S0.5	0.500	**0.499**	**0.499**	0.501	0.500	**0.499**
RESIZE	S0.75	0.500	**0.499**	**0.499**	0.500	0.500	**0.499**

4.2.7 Chaotic Parameter Ranges and Validation

For the Logistic map $x_{n+1} = r\,x_n(1 - x_n)$, we select $r = 3.8$ with $x_0 \in (0,1)$, which lies in the well-known chaotic regime (approximately $r \gtrsim 3.57$). We validated chaos using the Lyapunov exponent criterion ($\lambda > 0$).

For our setting ($r = 3.8$, $x_0 = 0.8$), the estimated Lyapunov exponent is $\lambda_{\text{logistic}} \approx 0.4299$, confirming chaotic behavior. For the Hénon map $x_{n+1} = 1 - ax_n^2 + y_n$, $y_{n+1} = bx_n$, we adopt the standard chaotic parameters $a = 1.4$ and $b = 0.3$ with $(x_0, y_0) \in (0, 1)$. Our validation yields a positive largest Lyapunov exponent $\lambda_{\text{henon}} \approx 0.4160$, confirming chaos under the adopted configuration, as shown in Fig. 11.

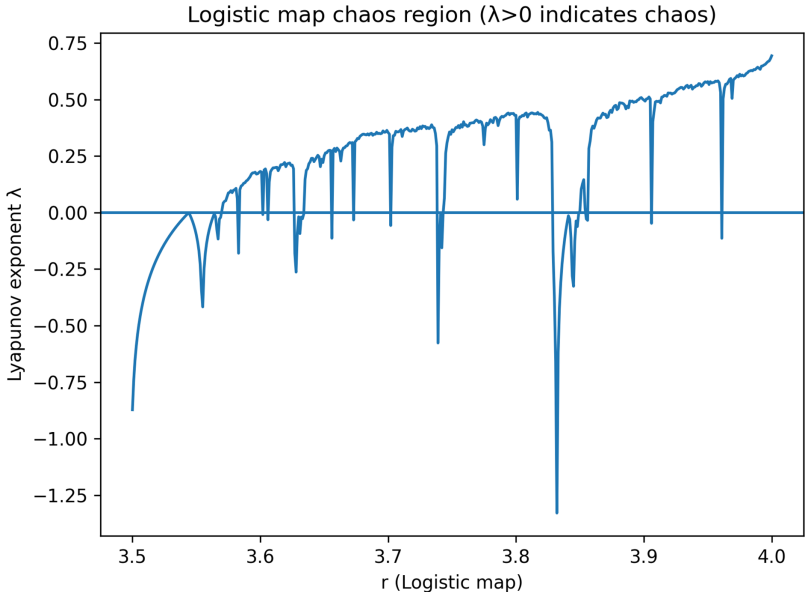


Figure 11: Lyapunov exponent of the logistic map vs. r. The selected $r = 3.8$ lies in a chaotic region ($\lambda > 0$).

4.2.8 Finite-Precision Clarification and Implementation Precision

Chaotic maps are known to be sensitive to numerical precision. We use IEEE-754 double-precision floating-point arithmetic (64-bit), which provides 52 effective mantissa bits per independent real-valued seed. For clarity, all chaotic-map computations and validations reported in this work were executed using IEEE-754 *double-precision* arithmetic (Python/NumPy default floating-point). In particular, our empirical validation computes the Lyapunov exponent for the Logistic map over the interval $r \in [3.5, 4.0]$ (where $\lambda > 0$ indicates chaotic behavior) and computes the largest Lyapunov exponent for the Hénon map under the parameters used in our pipeline (e.g., $a = 1.4$, $b = 0.3$), yielding a positive largest exponent (e.g., $\lambda \approx 0.416$), which confirms operation in a chaotic regime. These validations are produced by a dedicated script and recorded as artifacts (figure and text log).

Lyapunov exponent (Logistic map). For $x_{n+1} = rx_n(1 - x_n)$, we estimate

$$\lambda \approx \frac{1}{T} \sum_{n=1}^{T} \ln \left| r\left(1 - 2x_n\right) \right|, \tag{9}$$

after discarding an initial burn-in period. A positive λ indicates chaos.

4.3 Computational Complexity Analysis of Embedding Stages

Analysis of the embedding stage runtime reveals a clear computational trade-off dictated by the choice of procedures. The initial First Pass (FP) pairing exhibits linear time complexity, $O(N_s)$, dependent only on the secret image size (N_s). The subsequent Chaotic Second Pass (CSP) introduces sorting to identify pixels for re-pairing, increasing complexity slightly to $O(N_s \times logN_s)$. Finally, the optional Tertiary Full Pass (TFP)

phase, designed for maximal quality enhancement via a full fixing procedure, incurs a significantly higher cost of $O(N_s \times N_c)$, dependent on both secret (N_s) and cover (N_c) image sizes due to its linear search through all cover blocks. Since the stages are executed sequentially (e.g., running TFP necessitates completing the First Pass and CSP), employing all three for optimal perceptual quality results in the highest runtime, dominated by the TFP complexity. This $O(N_s \times N_c)$ cost is particularly pronounced when utilizing high-resolution cover images (large N_c). This establishes an inverse relationship (trade-off) between computational efficiency and achievable image fidelity, allowing the proposed method to be configured to prioritize either speed (by omitting CSP or TFP) or quality (by including all phases) based on specific application requirements and constraints. Measured execution times per configuration are reported in Tables 7 and 8.

4.4 Steganalysis Attacks Resistance Analysis

The security of our proposed image steganography scheme is evaluated against common steganalysis attacks, considering the cumulative effects of chaos-based permutation and encryption, chaotic FP LSB mapping, and embedding optimization (CSP and TFP).

Visual Attacks: The proposed method exhibits high resistance to visual attacks. Standard LSB substitution introduces minimal perceptual distortion. This is further enhanced by the Chaotic Second Pass (CSP) and Tertiary Full Pass (TFP) embedding phases, which actively seek embedding locations that minimize bit flips by optimizing for higher matching scores, thereby preserving the cover image's visual fidelity. The initial permutation and encryption phases solely affect the secret data and do not alter the appearance of the cover image.

Statistical Attacks (Spatial Domain): Resistance to traditional statistical attacks is significantly improved compared to naïve LSB methods.

- Chi-Square and PoV Attacks: These attacks often rely on deviations from expected pixel value frequencies or LSB pair statistics caused by sequential embedding. The utilization of Hénon and Logistic maps to pseudo-randomly distribute embedding blocks across the cover image disrupts this sequential assumption, rendering basic Chi-Square and Pairs of Values (PoV) analyses less effective.
- RS Steganalysis: Regular Singular pairs analysis (RS) detects embedding by measuring changes in image regularity/smoothness based on pixel group flippability. While chaotic selection provides some randomization, the score optimization in CSP and TFP directly reduces the number of LSB flips. Minimizing flips inherently decreases the disturbance to image regularity statistics, thus providing resistance against RS attacks, superior to standard LSB replacement or the First Pass embedding alone.
- SPA/PDH Attacks: The Sample Pair Analysis (SPA) and Pixel Difference Histogram (PDH) methods analyze correlations between adjacent pixel values or differences. The proposed chaotic-based method mitigates assumptions based on strict adjacency, and the flip reduction by CSP/TFP helps preserve original difference statistics. Therefore, it provides good resistance by reducing the subtle effects of LSB modification that can be detected by these attacks.

4.4.1 Undetectability Evaluation: SRM-Based Steganalysis Baseline (Small-Scale)

To provide an initial undetectability baseline, we evaluated a classical Spatial Rich Model (SRM) feature pipeline on a balanced set of Portable Gray Map (PGM) images, producing a feature matrix of size $78 \times 34,671$ (78 samples, 34,671 SRM features).

SRM feature dimensionality. The SRM extractor used in our MATLAB pipeline corresponds to the standard SRM rich model (106 submodels), whose total feature dimensionality is $D = 34,671$ as defined by the SRM implementation. In our dataset construction script, we also verify this programmatically by extracting

SRM features for a sample image, flattening the SRM-struct output into a single vector, and setting D = numel$(\cdot)$, which is then used to allocate the feature matrix $X \in \mathbb{R}^{2N \times D}$.

The dataset contains 39 *cover* and 39 *stego* images (stego images generated using our embedding procedure under the same experimental settings used in the main evaluation).

Protocol. To avoid data leakage, we used stratified 5-fold cross-validation with a fixed Random number generator (RNG) seed, set in MATLAB as `rng(7)`.

Run count. This small-scale baseline reports one stratified 5-fold CV run under a fixed seed (`rng(7)`), i.e., five held-out evaluations (one per fold).

Pairing integrity (anti-mixing). Before feature extraction, we enforce strict one-to-one pairing between cover and stego samples by matching filenames. The cover file list is sorted deterministically, and the pipeline verifies that each cover has a corresponding stego image with the exact same filename; otherwise the run aborts. SRM features for cover and stego are then extracted using this shared ordered list, ensuring row-aligned feature matrices and preventing any cover/stego mixing.

All experiments were run in MATLAB R2025b on a 13th Gen Intel Core i7-13650HX machine with 24 GB RAM. Within each fold, features were standardized using the mean and standard deviation computed on the training split only, and the same transformation was applied to the corresponding test split.

Leakage control (implementation-aligned). We enforce a strict no-leakage protocol in MATLAB. Stratified 5-fold splits are created by shuffling cover and stego indices independently under a fixed seed (`rng(7)`) and distributing each class evenly across folds. Within each outer fold, feature standardization uses training statistics only: the mean and standard deviation are computed on the outer-training split and then applied to both outer-training and outer-test data. The regularization strength λ is selected via an inner validation split constructed exclusively from the outer-training data (80/20 inner train/validation). After selecting λ, we retrain on the full outer-training split and evaluate only on the held-out outer-test split.

Classifier. As a lightweight baseline classifier, we trained an L_2-regularized logistic regression model (implemented without external toolboxes). Hyperparameters (including the regularization strength λ, and the decision direction when needed) were selected using an inner validation procedure restricted to the training portion of each outer fold.

Results. Across the five folds, the mean accuracy was 71.61% ± 10.11% (standard deviation across folds). Aggregating predictions over all held-out folds yields aggregated confusion counts: True Positives (TP) = 31, True Negatives (TN) = 25, False Positives (FP) = 14, and False Negatives (FN) = 8. The pooled Receiver Operating Characteristic (ROC) analysis resulted in Area Under the Curve (AUC) = 0.6568, indicating moderate separability between cover and stego samples in this small-scale setting. The SRM baseline uses a fixed payload setting for the PGM subset (e.g., 0.2 bpp in the dataset generation script), while maximum-payload behavior is assessed separately in the main payload-scaling experiments (up to 180 × 180, 32,400 B). This experiment is intended as a controlled baseline to validate the end-to-end evaluation pipeline (feature extraction → splitting → training → testing). Stronger conclusions on generalization require re-evaluation on larger and more diverse datasets. Notably, the confusion counts correspond to the selected operating point (fold-wise threshold), whereas the ROC curve is computed by sweeping the decision threshold over the pooled continuous scores from all held-out folds (Algorithm 10), shown in Fig. 12.

ROC/AUC computation. To compute the ROC curve, we pool the continuous decision scores obtained on the held-out *outer-test* split of each fold and aggregate them across the five folds (see Algorithm 10). The ROC curve is then obtained by sweeping the decision threshold over these pooled scores, and the AUC is computed using trapezoidal integration of the (FPR, TPR) curve. This is implemented directly in MATLAB

(no external toolboxes). In contrast, the reported confusion counts correspond to the selected operating point defined by the fold-wise threshold chosen during inner validation.

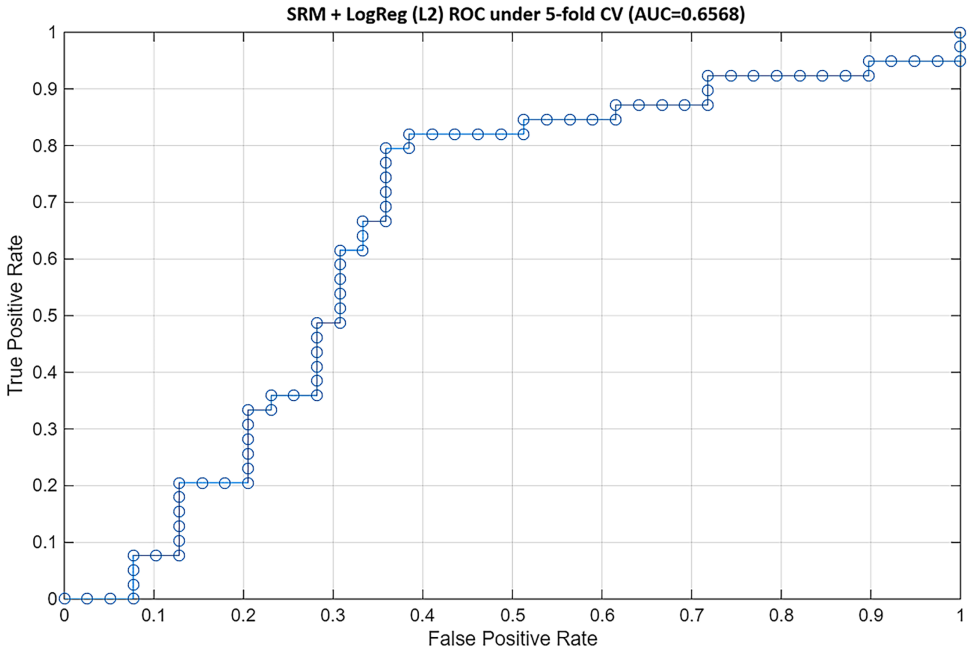


Figure 12: ROC curve of the SRM-based steganalysis baseline using ℓ_2-regularized logistic regression under stratified 5-fold cross-validation. The pooled ROC-AUC is 0.6568.

Algorithm 10: Pooled ROC/AUC from outer-fold held-out scores (SRM Baseline)

Require: Feature matrix X, labels y, number of folds $K = 5$
Ensure: Pooled ROC curve $(\mathrm{FPR}, \mathrm{TPR})$ and AUC
1: Initialize empty vectors: $S \leftarrow [\,]$ (scores), $L \leftarrow [\,]$ (labels)
2: **for** $k = 1$ to K **do**
3: Split indices into outer-train tr_k and outer-test te_k (stratified)
4: Compute (μ, σ) from $X(tr_k, :)$ only; standardize $X(tr_k, :)$, $X(te_k, :)$ using (μ, σ)
5: Train logistic regression on standardized $X(tr_k, :)$, $y(tr_k)$ (with λ selected on an inner split of tr_k)
6: Compute continuous held-out scores $s \leftarrow f(X(te_k, :))$ (e.g., sigmoid output)
7: Append: $S \leftarrow [S; s]$, $L \leftarrow [L; y(te_k)]$
8: **end for**
9: Compute ROC by sweeping threshold over pooled S vs. L; compute AUC by trapezoidal integration
10: **return** $(\mathrm{FPR}, \mathrm{TPR})$, AUC

4.4.2 Deep-Learning-Based Steganalysis Baseline (Small-Scale)

To complement the SRM-based detector with a learning-based reference, we conducted a small-scale CNN steganalysis experiment using a lightweight residual-oriented architecture (fixed high-pass prefilter followed by a shallow convolutional trunk). We followed a strict holdout protocol aligned with our dataset splits: the detector is trained on TRN, the best epoch is selected on VAL, and final performance is reported on the unseen TST split (balanced cover/stego, filename-paired (one-to-one) and balanced cover/stego splits). The experiment uses the same stego generation setting as in our SRM undetectability baseline (same embedding configuration and payload setting for the evaluated subset). Using crop size 256 and training

for 10 epochs (batch size 32), the CNN achieved a test AUC of 0.5001 and test accuracy of 50.0%, i.e., performance close to random guessing in our embedding settings. This provides an explicit deep-learning reference baseline to contextualize undetectability in this controlled setup. While we did not reproduce a full SRNet training pipeline in this setup, we explicitly included a lightweight CNN holdout baseline as a controlled deep-learning reference, as shown in Fig. 13.

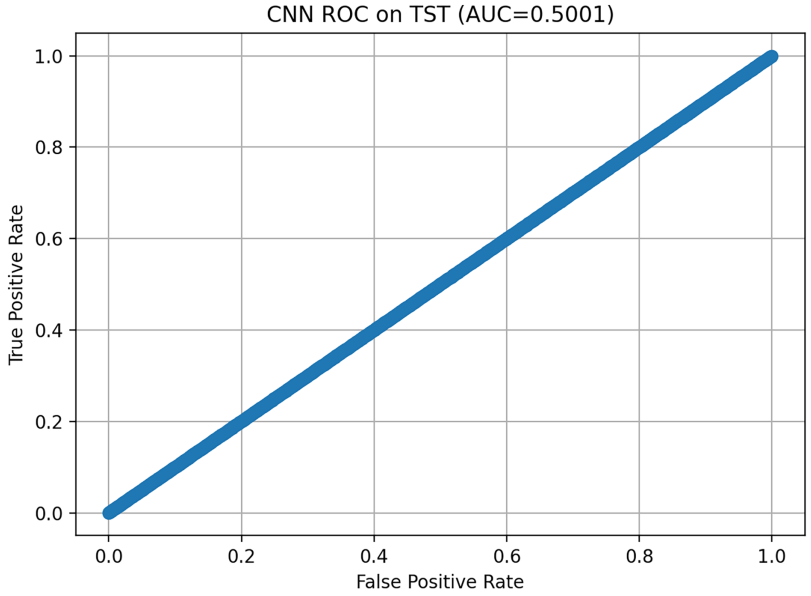


Figure 13: ROC curve of the lightweight CNN steganalysis baseline under the holdout protocol (Train on TRN, select on VAL, report on TST). The test AUC is 0.5001, indicating near-random separability for the considered embedding setting.

Scope note. CNN steganalyzers such as SRNet-style architectures typically require (i) training and evaluation on standardized steganalysis benchmarks (e.g., BOSSBase/ALASKA-like protocols) to ensure comparability, and (ii) substantially larger-scale training with carefully tuned schedules and preprocessing to obtain stable, reproducible performance. Reproducing a comparable SRNet-level evaluation (including cross-dataset testing) is therefore beyond the computational and dataset scope of this paper. We plan this as a separate extended study on standard benchmarks and cross-dataset protocols.

4.5 Analysis of Brute Force Attack Resilience

In our threat model, the stego image is transmitted over a public channel, while the metadata (seeds and address-selection information) is transmitted via a secure side channel. Without the metadata, a brute-force adversary must jointly guess the keystream seeds and the block-addressing sequence, which becomes computationally prohibitive as the payload and image size grow.

Key space estimation. In deployment, the secret key comprises the chaotic initial conditions used for keystream generation and the embedding address-selection sequence (and optionally the map parameters). Beyond the chaotic seed space, the block-wise addressing stage introduces a combinatorial selection space by choosing M distinct cover blocks from N available blocks, yielding $\binom{N}{M}$ possibilities. For the largest evaluated payload, the addressing contribution is: $log_2\binom{N}{M}$. For reproducibility in our experiments, we fix representative parameters (Table 2); however, in practice these values can be treated as session-specific secrets.

Finite-precision note. Since chaotic maps are realized under finite precision, we use double-precision arithmetic and validated chaotic ranges; optionally, a burn-in stage discards initial iterations prior to keystream sampling.

4.5.1 Explicit Key-Space Lower Bound (in Bits) and Security Thresholds

To express the key space in cryptographic terms, we derive a conservative lower bound from the *actual* confidentiality-critical parameters used in our implementation. In the encryption stage, a keystream is generated by the Hénon map using two real-valued seeds (x_0, y_0) through `gen_key(size, x0, y0)`, and ciphertext pixels are produced by XOR between the permuted plaintext and the generated keystream. Thus, the secret key material that controls decryption consists of the pair (x_0, y_0) used for keystream generation.

Assuming IEEE-754 double-precision arithmetic, each independent real-valued seed provides approximately 52 effective mantissa bits. Therefore, a conservative lower bound on the effective key space is:

$$\log_2 |\mathcal{K}_{enc}| \gtrsim 52\, n_s, \tag{10}$$

where n_s is the number of independent secret double-precision parameters. In our current implementation, $n_s = 2$ for the encryption keystream seeds (x_0, y_0), yielding:

$$\log_2 |\mathcal{K}_{enc}| \gtrsim 52 \times 2 \approx 104 \text{ bits}. \tag{11}$$

Concrete size. This corresponds to at least $|\mathcal{K}_{enc}| \gtrsim 2^{104}$ distinct encryption keys under the double-precision seed model, which is below the commonly cited 2^{128} brute-force threshold.

Meeting 128-bit thresholds. While 2^{104} offers substantial practical resistance to brute-force search, meeting the widely used 128-bit benchmark can be achieved *as a hardening extension* by adopting a 128-bit master key and deterministically mapping it to chaotic seeds (e.g., via a standard key-derivation function).

Addressing space vs. key space. The addressing list P (used for extraction) is treated as operational side-information rather than confidentiality-critical key material; thus it is not counted in $|\mathcal{K}_{enc}|$. If one reports the theoretical diversity of possible embedding patterns, it is upper-bounded by $\binom{N'_c}{N_s}$, but this term represents output-pattern diversity, not an independent cryptographic key component.

Meeting 128-bit thresholds. Achieving ≥ 128 bits would require at least three independent secret parameters (i.e., $n_s \geq 3$) or, alternatively, adopting a 128-bit master key and deterministically mapping it to chaotic seeds (e.g., via a standard key-derivation function), which we identify as a straightforward hardening extension.

5 Conclusion and Future work

This paper presented a hybrid chaos-based image security framework for the secure transmission of secret images. The methodology follows a two-phase protocol: (i) a hybrid permutation stage integrating Fisher-Yates with the Logistic map, and (ii) a Hénon-map-driven PRNG stream-cipher encryption stage.

A key contribution lies in the three-stage, configurable embedding strategy (FP, CSP, and TFP). Specifically, embedding operates at the byte level, where each secret byte is mapped into an 8-pixel cover block (one LSB per pixel). CSP and TFP then refine block placements by maximizing bit matching, thereby reducing the expected number of LSB modifications. Experimental results and the ablation analysis confirm that the full FP + CSP + TFP execution achieves the best imperceptibility (PSNR/SSIM/MSE) with reduced

embedded-region bit flips, at the cost of increased runtime, while also enabling practical trade-offs when faster configurations are desired.

The current framework remains constrained by the inherent fragility of spatial-domain LSB embedding under lossy or resampling operations, and by the limited scale of the present undetectability evaluation (classical tests, a small-scale SRM baseline, and a lightweight CNN holdout reference rather than a state-of-the-art steganalyzer).

Future work will prioritize improving robustness under lossy channels, expanding undetectability evaluation to stronger learning-based baselines (e.g., SRNet-like) under standardized within-/cross-dataset protocols and established benchmarks, optimizing runtime through reduced candidate search while preserving fidelity, and extending the approach to color images with channel-aware or transform-domain variants.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Islam T. Almalkawi: Conceptualization, methodology, overall supervision, and project administration; coordinated the revision process; writing: review & editing; corresponding author responsibilities. Samer Khasawneh: Methodology refinement, experimental design review, and validation; writing: review & editing. Hamza M. Alkhatib: Software implementation, experiments execution, data preparation/curation, and results documentation; writing: original draft (technical sections). Sabya Shtaiwi: Major contribution to the revision and resubmission; performed additional analyses/experiments requested during peer review; prepared and consolidated the complete point-by-point rebuttal and revision log; writing: review & editing. Rami Halloush: Technical review of the approach and results, supported interpretation and presentation improvements; writing: review & editing. Manel Guerrero Zapata: Formal analysis and critical technical feedback; contributed to strengthening the evaluation narrative; writing: review & editing. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The cover images used in this study were obtained from the publicly available USC-SIPI Image Database [39]. The stego images and related experimental outputs were generated during the execution of the proposed embedding algorithm described in this manuscript. Therefore, the experimental data can be reproduced from the USC-SIPI dataset and the methodology reported in this paper.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lorenzelli F. The essence of chaos. Raton, FL, USA: CRC Press; 2014.
2. Zolfaghari B, Koshiba T. Chaotic image encryption: state-of-the-art, ecosystem, and future roadmap. App Syst Innovat. 2022;5(3):57. doi:10.3390/asi5030057.
3. Zaidan BB, Zaidan AA, Al-Frajat AK, Jalab HA. On the differences between hiding information and cryptography techniques: an overview. J Appl Sci. 2010;10(15):1650–5. doi:10.3923/jas.2010.1650.1655.
4. Sheela S, Sathyanarayana SV. Application of chaos theory in data security-a survey. ACCENTS Trans Inform Secur. 2017;2(5):1–15. doi:10.19101/tis.2017.25001.
5. Morkel T, Eloff JHP, Olivier MS. An overview of image steganography. In: Proceedings of the Fifth Annual Information Security South Africa Conference; 2005 Jun 29–Jul 1; Balalaika Hotel, Sandton, South Africa. p. 1–11.
6. Singh AK, Singh J, Singh HV. Steganography in images using LSB technique. Int J Latest Tren Eng Technol (IJLTET). 2015;5(1):426–30.

7. Zaidan AA, Zaidan BB, Al-Frajat AK, Jalab HA. Investigate the capability of applying hidden data in text file: an overview. J Appl Sci. 2010;10(17):1916–22. doi:10.3923/jas.2010.1916.1922.
8. Duan X, Nao L, Mengxiao G, Yue D, Xie Z, Ma Y, et al. High-capacity image steganography based on improved FC-DenseNet. IEEE Access. 2020;8:170174–82. doi:10.1109/access.2020.3024193.
9. Konyar MZ, Solak S. Efficient data hiding method for videos based on adaptive inverted LSB332 and secure frame selection with enhanced Vigenere cipher. J Inf Secur Appl. 2021;63(1):103037. doi:10.1016/j.jisa.2021.103037.
10. Reddy VL, Subramanyam A, Reddy PC. Implementation of LSB steganography and its evaluation for various file formats. Int J Adv Netw Appl. 2011;2(5):868–72.
11. Solak S, Altınışık U. Image steganography based on LSB substitution and encryption method: adaptive LSB+ 3. J Electron Imaging. 2019;28(4):043025. doi:10.1117/1.jei.28.4.043025.
12. Sahu AK, Sahu M. Digital image steganography and steganalysis: a journey of the past three decades. Open Comput Sci. 2020;10(1):296–342.
13. Yadav P, Dutta M. A overview of various steganographic domains and its applications. Int J Eng Trends Technol. 2017;52(3):137–41.
14. Abdulwahed MN. An effective and secure digital image steganography scheme using two random function and chaotic map. J Theor Appl Inform Technol. 2020;98(1):96–105.
15. Hussain M, Riaz Q, Saleem S, Ghafoor A, Jung KH. Enhanced adaptive data hiding method using LSB and pixel value differencing. Multim Tools Appl. 2021;80(23):20381–401. doi:10.1007/s11042-022-14326-5.
16. Durafe A, Patidar V. Image steganography using fractal cover and combined chaos-DNA based encryption. Ann Data Sci. 2024;11(3):855–85. doi:10.1007/s40745-022-00457-x.
17. Manzar MA, Al-Ahmadi S, Hazzazi MM. A novel chaotic steganography method with three approaches for color and grayscale images based on FIS and DCT with flexible capacity. Multim Tools Appl. 2020;79(19):13693–724. doi:10.1007/s11042-019-08415-1.
18. Huo L, Chen R, Wei J, Huang L. A high-capacity and high-security image steganography network based on chaotic mapping and generative adversarial networks. Appl Sci. 2024;14(3):1225. doi:10.3390/app14031225.
19. Almalkawi IT, Halloush R, Alsarhan A, Al-Dubai A, Al-karaki JN. A lightweight and efficient digital image encryption using hybrid chaotic systems for wireless network applications. J Inf Secur Appl. 2019;49(7):102384. doi:10.1016/j.jisa.2019.102384.
20. Jaradat A, Taqieddin E, Mowafi M. A high-capacity image steganography method using chaotic particle swarm optimization. Secur Commun Netw. 2021;2021:6679284. doi:10.1155/2021/6679284.
21. Rustad S, Syukur A, Andono PN. Inverted LSB image steganography using adaptive pattern to improve imperceptibility. J King Saud Univ-Comput Inform Sci. 2022;34(6):3559–68. doi:10.1016/j.jksuci.2020.12.017.
22. Setiadi DRIM, Rustad S, Andono PN, Shidik GF. Graded fuzzy edge detection for imperceptibility optimization of image steganography. Imag Sci J. 2024;72(6):693–705. doi:10.1080/13682199.2023.2219880.
23. Ye H, Su K, Cheng X, Huang S. Research on reversible image steganography of encrypted image based on image interpolation and difference histogram shift. IET Image Process. 2022;16(7):1959–72. doi:10.1049/ipr2.12461.
24. Sahu AK. A logistic map based blind and fragile watermarking for tamper detection and localization in images. J Ambient Intell Human Comput. 2022;13(8):3869–81. doi:10.1007/s12652-021-03365-9.
25. Setiadi DRIM, Ghosal SK, Sahu AK. AI-powered steganography: advances in image, linguistic, and 3D mesh data hiding—a survey. J Future Artif Intell Technol. 2025;2(1):1–23.
26. Luo H, Ge B. Image encryption based on Hénon chaotic system with nonlinear term. Multim Tools Appl. 2019;78(24):34323–52. doi:10.1007/s11042-019-08072-4.
27. Aparna H, Madhumitha J. Combined image encryption and steganography technique for enhanced security using multiple chaotic maps. Comput Elect Eng. 2023;110(1):108824. doi:10.1016/j.compeleceng.2023.108824.
28. Lin Y, Yang Y, Li P. Development and future of compression-combined digital image encryption: a literature review. Digit Signal Process. 2025;158(2):104908. doi:10.1016/j.dsp.2024.104908.
29. Khalil N, Sarhan A, Alshewimy MA. A secure image steganography based on LSB technique and 2D chaotic maps. Comput Elect Eng. 2024;119:109566. doi:10.1016/j.compeleceng.2024.109566.

30. Abdulhammed OY. A robust image steganography based on a novel technique by using improved DNA and modified chaotic approach. J Supercomput. 2024;80(1):226–48. doi:10.1007/s11227-023-05459-x.
31. Eberl M. Fisher–Yates shuffle. Archive of Formal Proofs [online]. 2016 Sep 19. Available from: https://isa-afp.org/entries/Fisher_Yates.html.
32. Phatak SC, Rao SS. Logistic map: a possible random-number generator. Phys Rev E. 1995;51(4):3670–8.
33. Weisstein EW. Hénon map;. from MathWorld–a wolfram web resource. [cited 2026 Mar 15]. Available from: https://mathworld.wolfram.com/HenonMap.html.
34. Al-Hyari A, Obimbo C, Abu-Faraj MM, Al-Taharwa I. Generating powerful encryption keys for image cryptography with chaotic maps by incorporating collatz conjecture. IEEE Access. 2024;12(5):4825–44. doi:10.1109/access.2024.3349470.
35. Riley J. Understanding metadata. Washington, DC, USA: National Information Standards Organization (NISO); 2017.
36. Heron S. Advanced encryption standard (AES). Netw Secur. 2009;2009(12):8–12.
37. Harn L, Lin C. Authenticated group key transfer protocol based on secret sharing. IEEE Trans Comput. 2010;59(6):842–6. doi:10.1109/tc.2010.40.
38. Gastegger M. A performance comparison of 7z/LZMA and 7z/bzip2/tar. Graz, Austria: Graz University of Technology; 2007.
39. University of Southern California, Signal and Image Processing Institute. The USC-SIPI image database. [cited 2025 Dec 2]. Available from: https://sipi.usc.edu/database/.
40. Zhang Y, Jiang J, Zha Y, Zhang H, Zhao S. Research on embedding capacity and efficiency of information hiding based on digital images. In: Proceedings of the 2013 International Conference on Computer Sciences and Applications (CSA 2013). Lancaster, PA, USA: DEStech Publications, Inc.; 2013. p. 441–5.
41. Fardo FA, Conforto VH, de Oliveira FC, Rodrigues PS. A formal evaluation of PSNR as quality measurement parameter for image segmentation algorithms. arXiv:1605.07116. 2016.
42. Schluchter MD. Mean square error. In: Wiley StatsRef: statistics reference online. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2014.
43. Nilsson J, Akenine-Möller T. Understanding SSIM. arXiv:2006.13846. 2020.
44. Wu Y, Noonan JP, Agaian S. NPCR and UACI randomness tests for image encryption. Cyber journals: multidisciplinary journals in science and technology. J Select Areas Telecommun (JSAT). 2011;1(2):31–8.