

A Finite-Precision Robust Crypto-Steganographic Scheme for Color Images Using Perturbed 2D-LSCM and Chaotic Pixel Permutation

Zaid Rajih Mohammed [1*], Mohammed Hayder Kadhim [2]
[1] Faculty of Medical Sciences, Jabir ibn Hayyan University for Medical and Pharmaceutical Sciences, Najaf, Iraq
[2] The College of Medicine, Jabir ibn Hayyan University for Medical and Pharmaceutical Sciences, Najaf, Iraq

* Corresponding Author: **Zaid Rajih Mohammed**

Article Info

ISSN (Online): 3107-6580
Volume: 02
Issue: 06
Received: 20-08-2026
Accepted: 22-09-2026
Published: 24-10-2026
Page No: 01-18

Abstract
Chaos-based crypto-steganographic methods for colour images often use a pseudo-random pixel-position generator and a chaotic keystream. However, their behaviour under finite-precision computation is not always examined. In this study, both components were evaluated and two major weaknesses were identified. The modular position generator used in an earlier method could address only 85.7% of the pixels in a 512×512 image. In addition, the 2D Logistic-Sine-Coupling Map (2D-LSCM) was found to produce short cycles for 8.4% of randomly selected keys when implemented with IEEE-754 double precision. To overcome these limitations, an improved crypto-steganographic scheme was developed for hiding arbitrary file types in colour images. SHA-256 was used to derive independent sub-keys from a password and a random 16-byte salt. The payload was encrypted using multi-level substitution and nonlinear S-box diffusion. A perturbed 2D-LSCM combined with AES-256 in counter mode was used to improve keystream security. The encrypted data were embedded using a chaotic Fisher-Yates pixel permutation, LSB matching, and OPAP. The proposed generator and ciphertext successfully passed all 11 verified NIST SP 800-22 statistical tests. The ciphertext entropy ranged from 7.9795 to 7.9990 bits/byte, while a one-bit change in the plaintext, password, or salt modified 99.54–99.68% of ciphertext bytes. Experiments on 1,800 colour images derived from DIV2K showed PSNR values decreasing from 65.9 dB at 0.1 bpp to 55.9 dB at 1 bpp, with a windowed SSIM of at least 0.9989. Chi-square, RS, and sample-pair steganalysis detected LSB replacement but not the proposed embedding on three of the four USC-SIPI covers.

DOI: https://doi.org/10.54660/.IJECA.2026.2.6.01-18

Keywords: steganography, cryptography, chaotic map, 2D-LSCM, LSB matching, arbitrary file hiding, colour image, steganalysis

1. Introduction

The volume of sensitive data exchanged over public networks, including medical records, legal documents, financial statements and executable software, continues to grow, and so does its exposure to interception. Cryptography protects the content of such data by rendering it unintelligible to anyone who lacks the key. The ciphertext itself, however, is conspicuous: its random-looking, high-entropy appearance signals that something valuable is being transmitted and may invite attack, blocking or coercion (Cheddad *et al*., 2010; Kadhim *et al*., 2019) [11, 21]. Steganography addresses this complementary problem by concealing the very existence of the communication inside an innocuous carrier, most commonly a digital image, so that an observer perceives nothing unusual (Subramanian *et al*., 2021) [32].

Neither technique is sufficient on its own. A steganographic system whose embedding rule becomes known exposes its payload directly, while an encrypted file sent in the open reveals that secret communication is taking place (Bailey and Curran, 2006) [6]. Encrypting the payload first and then embedding the ciphertext yields two independent barriers: an adversary must first detect and extract the hidden data and then break the cipher. Spatial-domain least-significant-bit (LSB) substitution remains the most widely used embedding primitive because it is simple, offers high capacity and allows exact recovery (Chan and Cheng, 2004) [10]. Sequential LSB embedding, however, leaves statistical traces that are exposed by classical attacks such as the chi-square attack (Westfeld and Pfitzmann, 2000) [35], RS analysis (Fridrich *et al*., 2001) [17] and sample-pair analysis (Dumitrescu *et al*., 2003) [13], as well as by rich-model and deep-learning detectors (Fridrich and Kodovsky, 2012; Boroumand *et al*., 2019) [9, 16]. Two countermeasures are therefore standard: selecting embedding positions pseudo-randomly under a secret key, and encrypting the payload so that the embedded bits are statistically indistinguishable from noise.

Research published between 2024 and 2026 has advanced this combination considerably. Chaotic maps are now routinely used to generate embedding positions and keystreams (Khalil *et al*., 2024; Dsa and Patro, 2026; Naveen *et al*., 2026) [12, 23, 26]; standard ciphers such as AES and Blowfish have been paired with adaptive or saliency-guided LSB embedding (Banoori *et al*., 2025; Aljughaiman and Alrawashdeh, 2025, 2026) [2, 3, 8]; and learning-based models select embedding regions that resist CNN-based steganalysis (Alrawashdeh *et al*., 2025; Sanjalawe *et al*., 2025) [4, 29]. A closer examination of these works (Section 2) nevertheless reveals three recurring limitations:

- **Payload restriction:** Most schemes are designed for one payload type, typically short text messages or grayscale secret images. They cannot carry an arbitrary file together with the metadata (length and type) the receiver needs to reconstruct it without side information.
- **Capacity-dependent reporting:** High PSNR values are frequently reported at very low payloads, often below 0.4 bits per pixel (bpp), or on grayscale covers. Such figures do not reflect performance at practical capacities in colour images, which makes cross-paper comparison misleading.
- **Incomplete security evaluation:** Several schemes rely on ad-hoc lightweight ciphers without key-space, key-sensitivity or differential (NPCR/UACI) analysis, contrary to the basic requirements established for chaos-based cryptosystems (Alvarez and Li, 2006) [5].

In earlier work, Shukur *et al*. (2018) [31] proposed a two-stage method that encrypts a file of any extension with a modified Playfair cipher built on a random 16×16 table and hides the result in pseudo-randomly selected pixels of a colour image. That method showed that DOCX, PDF, EXE, MP3, RAR and other files could be encrypted to near-maximal entropy and hidden with a PSNR of about 47 dB at roughly 3.4 bpp. It left several parts unspecified, however: the decryption and extraction procedures, the hash and state-update functions, and key management. Its cipher is a pure byte substitution without diffusion, its modular position generator can enter a deadlock, and no security or steganalysis evaluation was reported.

This paper revisits and substantially extends that method. Its contributions are of two kinds: measurements that expose defects in components widely used by this family of schemes, and a scheme that removes them.

- **Two measured defects:** We show, by bisection on the number of requested pixels, that the modular position generator of Shukur *et al*. (2018) [31] stalls after at most 85.7% of a 512×512 image, and, over 10,000 random keys, that the unperturbed 2D-LSCM falls into short cycles in double precision for 8.4% of keys. Both effects are silent: the programs run and produce output, but the positions stop or the keystream repeats. We quantify how much of each fix is needed: a 2^-24 perturbation removes every cycle, but the perturbed map alone still fails part of the NIST SP 800-22 suite, and only the hedged generator passes it.
- **Arbitrary-file support with salted keys:** A complete, blindly recoverable scheme for hiding files of any type in colour images. A per-message random salt and an encrypted self-describing header (length, extension, depth and integrity checksum) enable exact extraction without side information, and make two messages under the same password cryptographically independent.
- **Substitution-diffusion cipher:** The multi-level substitution of the original cipher is retained and followed by a forward and a backward non-linear diffusion pass through the key-dependent S-box, so that a single-bit change anywhere in the plaintext, password or salt alters the entire ciphertext; the counter streams run on AES-256-CTR, which makes the cipher several times faster than with SHA-256.
- **Collision-free chaotic embedding:** Fisher-Yates permutation of all pixels driven by the protected generator, with LSB matching at one bit per channel and OPAP at higher depths, the depth being selected automatically from the file size.
- **Large-scale evaluation:** 1,800 colour covers from DIV2K and four USC-SIPI images; NIST SP 800-22 on the generator and on ciphertexts of an all-zero file; chosen-plaintext tests; key and salt sensitivity; chi-square, RS and sample-pair steganalysis; execution time; and a comparison with recent schemes against the theoretical PSNR-payload curves, which shows that most reported PSNR differences are explained by payload and cover type rather than by the embedding method.

The rest of the paper is organised as follows. Section 2 reviews related work. Section 3 describes the proposed scheme. Section 4 presents the experimental setup and results. Section 5 analyses security, and Section 6 concludes the paper.

2. Related Work

Image steganography has been surveyed extensively (Cheddad *et al*., 2010; Kadhim *et al*., 2019; Subramanian *et al*., 2021) [11, 21, 32]. Classical spatial-domain methods include

LSB substitution with optimal pixel adjustment (Chan and Cheng, 2004) [10] and pixel-value differencing (Wu and Tsai, 2003) [36]; content-adaptive methods such as S-UNIWARD minimise a distortion function to place changes in textured regions (Holub *et al*., 2014) [19]; and deep networks can hide whole images inside covers (Baluja, 2017) [7]. This section focuses on crypto-steganographic schemes published in Q1/Q2 journals between 2024 and 2026, grouped by their main design idea.

2.1. LSB Embedding with Conventional or Lightweight Encryption

Rahman *et al*. (2025) [27] combined a multi-level encryption algorithm (XOR operations and circular bit shifts) with LSB substitution of up to four bits, magic-matrix shuffling of blue-channel blocks, transposition and flipping. Tested on 165 RGB images of four sizes and formats, the method achieved an average PSNR of 81.32 dB on 512×512 covers with a 14 KB text payload and showed strong resistance to RS analysis but weak resistance to pixel-difference-histogram (PDH) analysis. Banoori *et al*. (2025) [8] replaced the lightweight cipher with AES-128 and embedded the ciphertext in the blue channel after magic-matrix shuffling, XOR and transposition; they reported an average PSNR of 80.49 dB against 70.22-77.46 dB for five earlier methods. Both schemes are restricted to text, and their very high PSNR values reflect small payloads relative to the cover size. Neither reports execution time.

2.2. Chaos-Based Crypto-Steganography

Chaotic maps are attractive because of their sensitivity to initial conditions and their low computational cost. Khalil *et al*. (2024) [23] used a hybrid of a one-dimensional and a two-dimensional composite chaotic map to generate the embedding position vector and optimised the positions to match cover bits with secret bits, supporting image, text and audio payloads and reporting average improvements of 4.76 dB in PSNR and 0.0292 in SSIM over compared methods. Dsa and Patro (2026) [12] merged encryption into embedding using two piecewise linear chaotic maps and in-place XOR, randomising both the payload reading order and the pixel positions, and used Huffman compression to raise capacity. Their scheme accepts TXT, PDF and DOC files, reaches a peak PSNR of 77.42 dB for small payloads and remains above 51 dB at a full capacity of 166,336 characters, has a key space exceeding 1.2483 × 2^199, completes in under 1.6 s, and passes RS, sample-pair and chi-square tests. It is the work closest to ours in scope. Naveen *et al*. (2026) [26] modified the Blowfish round function with genetic crossover and mutation and embedded the ciphertext with a chaotic-quantisation LSB rule driven by a logistic map ($\mu = 3.6$); with 4 KB text payloads in colour PNG covers they reported PSNR of 66.53-71.86 dB and SSIM above 0.9999, supported by RS analysis. Aljughaiman and Alrawashdeh (2026) [3] combined Blowfish with priority-guided LSB embedding driven by fused entropy and Laplacian maps on grayscale BOSSbase covers; PSNR decreased from 61.28 dB at 0.1 bpp to 51.21 dB at 1.0 bpp, and Xu-Net and Ye-Net detectors performed near chance level (AUC 0.49-0.57).

2.3. Adaptive and Learning-Assisted Schemes

Aljughaiman and Alrawashdeh (2025) [2] protected secret images with AES-128-GCM and RSA-OAEP key wrapping and used ant colony optimisation guided by saliency and gradient maps to disperse LSB changes. At 0.045-0.397 bpp on grayscale covers they obtained 55.17-64.5 dB PSNR with SSIM above 0.9978, although embedding could take up to 15.22 s. Alrawashdeh *et al*. (2025) [4] used holistically-nested edge detection and a CNN to predict attention maps and allocated one to four LSBs per pixel accordingly, with a logistic-map XOR cipher; they reported 60.72-61.20 dB at 0.1 bpp, 55.16 dB at 0.397 bpp and detector AUCs of 0.47-0.53. Sanjalawe *et al*. (2025) [29] layered Huffman-coded LSB embedding with a deep encoder-decoder on 64×64 colour images; the LSB layer reached 61.80-62.40 dB at about 2.3 bpp, but the deep layer only 18.31-26.39 dB, and Huffman coding provides compression rather than confidentiality. In the transform domain, Kumar *et al*. (2026) [24] embedded a secret image encrypted by arithmetic, geometric and harmonic progressions into contourlet coefficients, obtaining an average PSNR of 45.5 dB and SSIM of 0.9837 with good robustness to noise, JPEG compression, rotation and cropping.

2.4. Summary and Research Gap

Table 1 summarises the reviewed works. Three observations motivate this paper. First, only Dsa and Patro (2026) [12] and, partially, Khalil *et al*. (2024) [23] accept payloads other than text or images, and none of the reviewed schemes documents how file type and length are conveyed for blind recovery of arbitrary files. Second, most reported PSNR values above 60 dB are obtained below 0.4 bpp or with small text payloads, so a fair comparison must be made at equal bpp. Third, schemes that use custom lightweight ciphers seldom report key sensitivity or NPCR/UACI, which are required to establish the strength of chaos-based encryption (Alvarez and Li, 2006; Wu *et al*., 2011) [5, 37]. The proposed scheme targets these three gaps while retaining the high capacity and simplicity of the original method (Shukur *et al*., 2018) [31].

Table 1: Comparison of recent crypto-steganographic schemes (2024-2026) with the baseline and proposed methods

Scheme	Encryption	Embedding	Cover	Payload	Any file	Capacity	PSNR (dB)	SSIM	Steganalysis
Khalil *et al*. (2024) [23]	1D+2D chaotic maps	Chaotic-position LSB	Colour	Image/text/audio	Partly	n/r	+4.76 vs. prior	+0.029 vs. prior	Statistical
Rahman *et al*. (2025) [27]	MLEA (XOR, shifts)	LSB ≤4 bits, magic matrix	Colour	Text	No	8-14 KB	71.8-84.1	≈0.999	RS, PDH
Banoori *et al*. (2025) [8]	AES-128	Blue-channel LSB, magic matrix	Colour	Text	No	2-12 KB	53.2-87.9	0.879-0.999	RS
Sanjalawe *et al*. (2025) [29]	Huffman (no cipher)	LSB + deep encoder	Colour 64×64	Text	No	2.0-2.35 bpp	61.8-62.4	≈0.9998	WOW, SRM, SRNet
Aljughaiman & Alrawashdeh (2025) [2]	AES-GCM + RSA-OAEP	ACO saliency LSB	Gray	Image	No	0.045-0.397 bpp	55.2-64.5	>0.9978	Xu-Net, Ye-Net
Alrawashdeh *et al*. (2025) [4]	Logistic XOR	CNN edge-adaptive LSB	Gray	Image	No	0.1-0.397 bpp	55.2-61.2	0.9995	Xu-Net, Ye-Net, RS
Aljughaiman & Alrawashdeh (2026) [3]	Blowfish	Priority-guided LSB	Gray	Image	No	0.1-1.0 bpp	51.2-61.3	0.997-1.0	Xu-Net, Ye-Net, RS
Naveen *et al*. (2026) [26]	Blowfish + GA	Chaotic-quant LSB	Colour	Text	No	4-12 KB	66.5-71.9	>0.9999	RS
Kumar *et al*. (2026) [24]	AP/GP/HP progressions	Contourlet	Gray	Image	No	Full image	45.5	0.9837	Robustness only
Dsa & Patro (2026) [12]	Dual PWLCM + XOR	Chaotic LSB + Huffman	Colour	TXT/PDF/DOC	Yes	≤166,336 chars	51-77.4	≈1.0	RS, SPA, χ^2
Shukur *et al*. (2018) [31] (baseline)	Modified Playfair	Modular random LSB	Colour	Any file	Yes	≈3.4 bpp	47.3-47.5	0.9997	None
Proposed	Salted SHA-256 keys + Playfair + 2D-LSCM/AES-CTR S-box diffusion	Chaotic permutation, LSB matching / OPAP	Colour	Any file	Yes	0.1-3 bpp (files 0.29-6.13 bpp)	49.3-65.9	≥0.9998	RS, SPA, χ^2

n/r = not reported in the abstract. Values are as reported by the respective authors; PSNR values are comparable only at equal payload.

3. Proposed Scheme

3.1. Overview

Fig. 1 shows the complete scheme. The sender and the receiver share only a password. At the sender, a fresh random salt is drawn and the password and salt are expanded into sub-keys (Section 3.2); the secret file, whatever its type, is read as a byte string and encrypted (Sections 3.3-3.4); an encrypted header describing the file is formed (Section 3.6); and salt, header and ciphertext are embedded along a key-dependent permutation of the cover pixels (Section 3.7). The stego image is saved as PNG. The receiver regenerates the permutation from the password, reads the salt, derives the remaining sub-keys, reads the header, extracts and decrypts the payload, verifies its checksum and restores the file with its original extension. No side information (file size, type, salt or embedding depth) is transmitted separately.

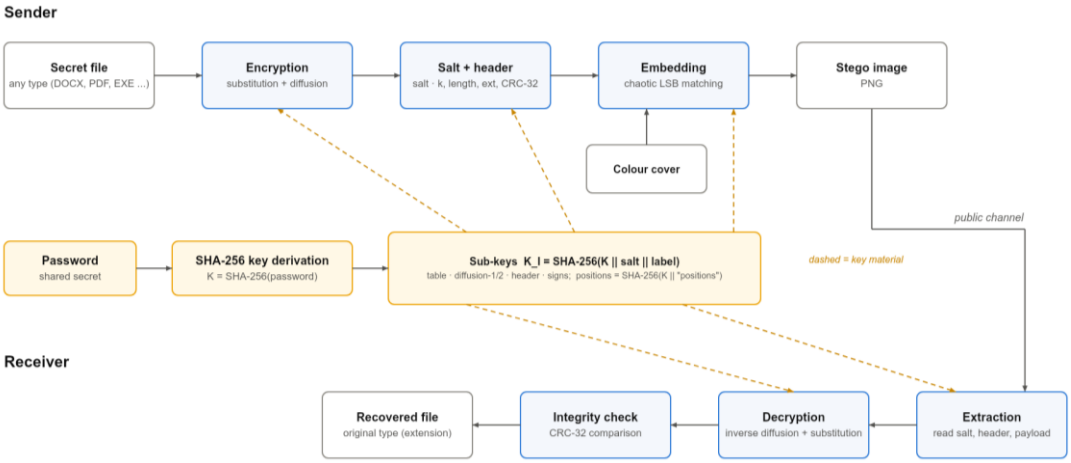


Fig 1: Block diagram of the proposed scheme (dashed arrows carry key material)

3.2. Key Derivation

Let P be the password and σ a 16-byte salt drawn from the operating system's cryptographic random generator for every message. The master key is K = SHA-256(P), and each component of the scheme uses its own sub-key

$$K_\ell = \text{SHA-256}(K \,\|\, \sigma \,\|\, \ell),\ \ell \in \{\text{table, diffusion-1, diffusion-2, header, embed-sign}\};\ K_{positions} = \text{SHA-256}(K \,\|\, \text{"positions"})$$

where ‖ denotes concatenation and ℓ is an ASCII label. K_table seeds the construction of the substitution structure of the original cipher (16×16 table, row and column alphabets and index hash); K_diffusion-1 and K_diffusion-2 seed the two diffusion keystreams; K_header masks the header; and K_embed-sign drives the ±1 decisions of LSB matching. K_positions, which seeds the pixel permutation, does not depend on the salt, because the receiver must know the permutation to find the salt. Because the sub-keys are outputs

of a hash function on distinct inputs, knowledge of one component (for example, the permutation recovered by an attacker who has the cover) reveals nothing about the others; and because the salt differs between messages, two files hidden under the same password are encrypted, masked and signed with unrelated keys.

3.3. The Chaotic Generator

All key-dependent sequences except the substitution table are produced by the 2D Logistic-Sine-coupling map (2D-LSCM) of Hua *et al*. (2018) [20]:

$$x_{i+1} = \sin(\pi(4\theta\, x_i\,(1 - x_i) + (1 - \theta)\sin(\pi\, y_i))),$$
$$y_{i+1} = \sin(\pi(4\theta\, y_i\,(1 - y_i) + (1 - \theta)\sin(\pi\, x_{i+1})))$$

which is chaotic for $\theta \in (0, 1)$ with a larger chaotic range than its constituent one-dimensional maps. The 32 bytes of a sub-key set the initial conditions: x_0 and y_0 from bytes 0-7, $\theta = 0.1 + 0.8 \cdot u$ from bytes 8-11, and bytes 12-27 add fine offsets to x_0 and y_0 so that every key bit influences the orbit. One thousand transient iterations are discarded.

Chaotic maps implemented in finite precision suffer dynamical degradation: their orbits eventually fall into cycles whose length can be far shorter than the theoretical behaviour suggests (Li *et al*., 2005; Alvarez and Li, 2006) [5, 25]. We measured this effect directly (Section 5.4): in IEEE-754 double precision the unperturbed 2D-LSCM collapsed into a short cycle within 20,000 iterations for 838 of 10,000 random keys (8.4%), in the worst case to a cycle of three values. A permutation driven by such a sequence is far from uniform. The generator is therefore protected by two mechanisms:

- **Anti-degradation perturbation:** before every iteration, x and y each receive an independent additive term $r \cdot 2^{-24}$, where $r \in [0, 1)$ is taken from a counter stream C_P = AES-256-CTR under the key SHA-256(K_ℓ ‖ "P") (Dworkin, 2001) [15]. The perturbation is too small to alter the map's local dynamics but prevents the orbit from closing into a cycle.
- **Hedged output:** every emitted value is combined with a second, independent counter stream C_O = AES-256-CTR under SHA-256(K_ℓ ‖ "O"): keystream bytes by XOR, and uniform numbers in [0, 1) by addition modulo 1. Each output sequence is therefore at least as unpredictable as AES-256 in counter mode even if the chaotic component were to degrade; the chaotic map contributes key-dependent non-linearity on top of this guarantee rather than being its sole source. Section 5.5 shows that the hedge is necessary: the perturbed map without it fails part of the NIST suite.

A keystream byte is formed as $(\lfloor 10^{10} \cdot |x| \rfloor \oplus \lfloor 10^{10} \cdot |y| \rfloor) \bmod 256$, and a uniform number as $(10^7 \cdot |x| + 10^5 \cdot |y|) \bmod 1$, each followed by the hedge. In an earlier version the two counter streams were built from SHA-256 blocks; AES-256-CTR gives the same construction with a standard stream cipher and, on processors with AES instructions, is several times faster (Section 4.5).

3.4. Encryption

Fig. 2 shows the cipher. It keeps the three-step multi-level substitution of Shukur *et al*. (2018) [31], now keyed by K_table, and adds a diffusion stage, which the original design lacked.

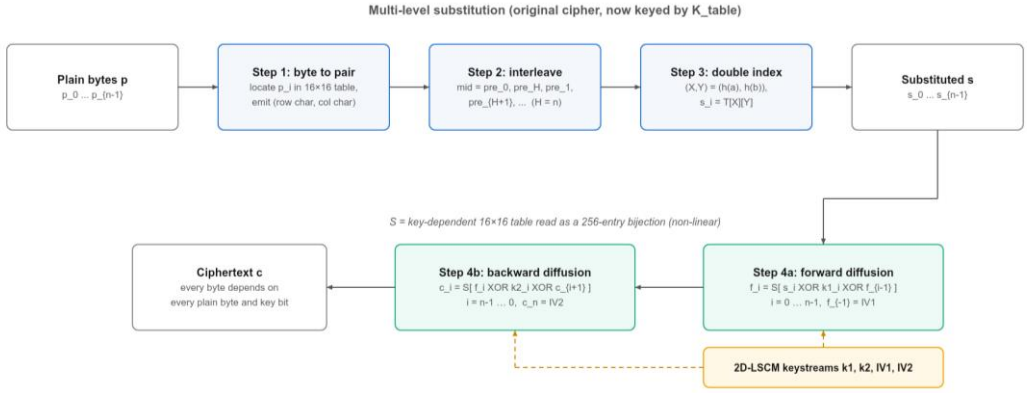


Fig 2: Encryption: multi-level substitution followed by two-pass non-linear diffusion

- **Step 1 (byte to pair):** a 16×16 table T holding a key-dependent permutation of 0-255, and two key-dependent orderings of the 16 letters A-P used as row and column alphabets, are built from K_table. Each plaintext byte p_i is located in T at (r, c) and replaced by the letter pair (rowChar[r], colChar[c]).
- **Step 2 (interleaving):** the 2n letters are rearranged by interleaving the first half with the second half, so that letters originating from different bytes are paired.
- **Step 3 (double indexing):** each new pair (a, b) is mapped by a key-dependent index hash h to (X, Y) = (h(a), h(b)) and replaced by s_i = T[X][Y].

Steps 1-3 form a keyed permutation of byte pairs; on their own they leave the ciphertext insensitive to small plaintext changes (NPCR ≈ 0%, Table 2). Step 4 introduces diffusion. Let S be the table T read as a bijection S(v) = T[v >> 4][v & 15] on 0-255, and let k1, k2 and the initial values IV1, IV2 be 2D-LSCM keystreams under K_diffusion-1 and K_diffusion-2:

Table 2: Encryption quality per file type (entropy in bits/byte; NPCR/UACI in %)

File	Size (KB)	H plain	H 2018	H proposed	Corr.	NPCR/UACI 2018 (bit)	NPCR/UACI proposed (bit)	NPCR/UACI proposed (key)	NPCR/UACI proposed (salt)	Enc./Dec. (ms)
ZIP	88.8	7.9956	7.9979	7.9982	0.00083	0.00/0.00	99.58/33.43	99.60/33.41	99.60/33.46	45/46
WAV	62.5	7.6177	7.9614	7.9972	0.00239	0.00/0.00	99.60/33.70	99.63/33.44	99.66/33.69	27/28
RTF	23.2	5.3498	7.4657	7.9928	-0.00312	0.01/0.00	99.54/33.43	99.66/33.20	99.66/33.56	10/11
PNG	58.8	7.9896	7.9966	7.9968	0.00860	0.00/0.00	99.61/33.36	99.60/33.43	99.61/33.40	25/25
BMP	75.1	7.6927	7.9830	7.9981	0.00320	0.00/0.00	99.64/33.59	99.61/33.47	99.63/33.48	31/32
EXE	196.0	6.3093	7.6294	7.9990	-0.00392	0.00/0.00	99.61/33.47	99.61/33.43	99.60/33.44	79/78
DOCX	19.1	7.8805	7.9683	7.9903	-0.00533	0.01/0.00	99.63/33.53	99.63/33.54	99.63/33.48	8/9
PDF	124.8	7.8059	7.9690	7.9985	0.00143	0.00/0.00	99.63/33.44	99.63/33.51	99.58/33.53	51/51
JPG	9.3	7.8747	7.9559	7.9795	0.01398	0.02/0.01	99.58/33.69	99.65/33.78	99.59/33.45	4/4
TXT	45.5	5.4359	7.3994	7.9959	-0.00409	0.00/0.00	99.62/33.59	99.61/33.44	99.59/33.59	19/19
XLSX	19.0	7.8632	7.9697	7.9910	0.00325	0.01/0.00	99.68/33.94	99.67/33.58	99.54/33.51	8/8

forward:

$$f_i = S[\, s_i \oplus k1_i \oplus f_{i-1} \,],\ i = 0 \ldots n-1,\ f_{-1} = IV1$$

backward:

$$c_i = S[\, f_i \oplus k2_i \oplus c_{i+1} \,],\ i = n-1 \ldots 0,\ c_n = IV2$$

Because S is non-linear, a change in any input byte propagates to every later byte in the forward pass and to every earlier byte in the backward pass, so each ciphertext byte depends on every plaintext byte and on every key bit. Algorithm 1 summarises the procedure.

Algorithm 1 Encrypt(p[0..n-1], P, salt)

1. K <- SHA-256(P); derive K_table, K_diffusion-1, K_diffusion-2 with salt
2. (T, rowChar, colChar, h) <- BuildTable(K_table)
3. s <- Substitute(p; T, rowChar, colChar, h) // Steps 1-3
4. k1 <- LSCM_bytes(K_diffusion-1, n+1); k2 <- LSCM_bytes(K_diffusion-2, n+1)
5. prev <- k1[n]
6. for i = 0 .. n-1: prev <- S[s[i] XOR k1[i] XOR prev]; f[i] <- prev
7. prev <- k2[n]
8. for i = n-1 .. 0: prev <- S[f[i] XOR k2[i] XOR prev]; c[i] <- prev
9. return c

3.5. Decryption

Decryption applies the inverse operations in reverse order. With S^{-1} the inverse table, the backward pass is undone first as $f_i = S^{-1}[c_i] \oplus k2_i \oplus c_{i+1}$ (with $c_n = IV2$), for $i = n-1$ down to 0, and then the forward pass as $s_i = S^{-1}[f_i] \oplus k1_i \oplus f_{i-1}$ (with $f_{-1} = IV1$), for $i = 0$ to $n-1$. Both loops use only ciphertext-side values that are already known, so decryption is exact. Finally Steps 3, 2 and 1 are inverted: each s_i is located in T to recover (X, Y), the letters $h^{-1}(X)$, $h^{-1}(Y)$ are restored, the interleaving is reversed, and each letter pair is mapped back to its table entry.

3.6. Salt and Self-Describing Header

The 16-byte salt is embedded in clear with one LSB per channel in the first 43 pixels of the permutation ($\lceil 16 \cdot 8/3 \rceil = 43$). The salt is not secret; its role is to make every message use fresh sub-keys, and without the password its position in the image is unknown. A 20-byte header follows: a 2-byte magic value "SQ", a version byte (2 in this format), the embedding depth k (1 byte), the file length (4 bytes), the file extension (8 bytes, zero-padded) and a CRC-32 of the plaintext (4 bytes). The header is XOR-masked with a 2D-LSCM keystream under the salted K_header and is embedded with one LSB per channel in the next 54 pixels ($\lceil 20 \cdot 8/3 \rceil = 54$), so salt and header together occupy 97 pixels and 36 bytes. At the receiver, a wrong password yields an invalid magic value or version and is rejected before any payload is read; a length that exceeds the capacity at the recorded depth is rejected as corruption; and after decryption the CRC-32 detects any modification of the stego image or payload. The header makes the scheme blind: the receiver needs neither the file size nor its type. Images produced by the earlier, unsalted format are recognised and rejected with an explicit message rather than a generic key error (Table 3).

Table 3: Functional and robustness checks

Check	Result
wrong key rejected	OK: wrong key or no hidden data (bad header)
tampered payload detected	OK: integrity check failed (CRC-32 mismatch): stego image modified or wrong key
oversize rejected before write	OK: secret too large: capacity 5998 bytes at 4 LSB/channel, need 127772
permutation 128x128 collision-free	OK
permutation 192x192 collision-free	OK
permutation 224x224 collision-free	OK
permutation 256x256 collision-free	OK
permutation 384x384 collision-free	OK
permutation 512x512 collision-free	OK
exact extraction embed=match k=1	OK
exact extraction embed=match k=2	OK
exact extraction embed=match k=3	OK
exact extraction embed=match k=4	OK
exact extraction embed=replace k=1	OK

Check	Result
exact extraction embed=replace k=2	OK
exact extraction embed=replace k=3	OK
exact extraction embed=replace k=4	OK
same file + password twice (random salt) -different ciphertext	OK: NPCR 99.60%, UACI 33.52%
format-v1 image rejected with clear message	OK: this stego image was made with format version 1 (before the per-message salt); extract it with the previous program version

3.7. Collision-Free Chaotic Embedding

The original method generated pixel coordinates with the modular recurrences i = (x·y + z) mod W and j = (x + y·z) mod H, discarding repeated positions and re-seeding after 100 consecutive collisions. This design has no guarantee of termination: as the fraction of used pixels grows, the probability of a new position vanishes and the generator stalls. We found that on a 512×512 image it cannot reach more than 85.7% of the pixels (Section 5.7). The proposed scheme instead builds a permutation π of all N = W·H pixel indices with the Fisher-Yates shuffle (Durstenfeld, 1964) [14], drawing each swap index j = ⌊u·(i+1)⌋ from the protected 2D-LSCM uniform sequence under K_positions. Every pixel appears exactly once, no collision test is needed, the cost is O(N) and the whole cover is usable (Fig. 3).



Fig 3: Chaotic pixel permutation (top) and layout of the embedded stream along π (bottom)

The salt occupies π[0..42] and the header π[43..96]. The ciphertext occupies π[97..] with k bits in each of the R, G and B channels of a pixel, where k is the smallest value in {1, 2, 3, 4} for which the file fits:

$$k = \min\{\, k \in \{1,2,3,4\} : n \le \lfloor (N - 97) \cdot 3k / 8 \rfloor \,\}$$

If no k satisfies the condition, the file is rejected before anything is written. The unused pixels are left untouched.
Two embedding rules are used, selected by k:

- **k = 1, LSB matching:** when the payload bit differs from the LSB of a channel value v, v is changed to v+1 or v−1 with the sign taken from a 2D-LSCM bit stream under K_embed-sign (v = 0 becomes 1 and v = 255 becomes 254). LSB matching produces the same LSB plane as LSB substitution but does not create the pairs-of-values asymmetry exploited by chi-square, RS and sample-pair analysis (Sharp, 2001) [30].
- **k ≥ 2, LSB substitution with OPAP:** the k LSBs are replaced and the optimal pixel adjustment process of Chan and Cheng (2004) [10] adds or subtracts 2^k when this reduces the error without altering the embedded bits. OPAP lowers the mean squared error; the LSB plane itself is still replaced.

Plain k-LSB replacement is retained as an ablation option to quantify the effect of the embedding rule (Sections 4.3 and 5.3).

3.8. Extraction

The receiver computes K_positions and π from the password, reads the 128-bit salt from the LSBs of π[0..42], derives the salted sub-keys, reads 160 bits from π[43..96], unmasks the header and checks the magic value, version, depth and length. It then reads n bytes from π[97..] at depth k, decrypts them (Section 3.5), recomputes the CRC-32 and compares it with the header value. On success, the file is written with its original extension; otherwise extraction fails with an explicit wrong-key or integrity error. Extraction reads only LSBs, so it is identical for LSB matching, OPAP and replacement.

4. Experimental Results

4.1. Setup

The scheme and the original method were implemented in JavaScript (Node.js 24) using only the standard crypto and zlib modules and the pngjs codec; the implementation is available from the authors. Experiments ran on an Intel Core i7-10700 (2.9 GHz) with 24 GB RAM under Windows 10; all timings are single-threaded. Two cover sets were used:

- **USC-SIPI:** four 512×512 24-bit colour images from the USC-SIPI database (USC-SIPI, 2024) [33]: Airplane (4.2.05), Baboon (4.2.03), Peppers (4.2.07) and Sailboat (4.2.06), converted losslessly to PNG. They are used for per-image tables, for hiding real files and for the classical steganalysis attacks. The Lena image was deliberately excluded, following the policy of several journals.

- **DIV2K:** the 900 high-resolution colour images of the DIV2K training and validation sets (Agustsson and Timofte, 2017) [1]. Each image was downscaled by area averaging so that its shorter side is 512 pixels, and two 512×512 crops were taken at the two ends of its longer side, giving 1,800 covers. The two crops of a source overlap by 256 pixels on average; for the 8 square sources they coincide, so the set contains 1792 distinct covers. DIV2K is used to measure imperceptibility at scale.

The secret files are eleven real files of different types: ZIP, WAV, RTF, PNG, BMP, EXE, DOCX, PDF, JPG, TXT and XLSX (9.3-196 KB). For controlled tests, random payloads were encrypted and embedded at 0.5, 1, 2 and 3 bpp on USC-SIPI and at 0.1, 0.2, 0.4 and 1 bpp on DIV2K, where bpp counts all embedded bits, including the 36-byte salt and header, per pixel of the whole image. The USC-SIPI results use the password "Zaid@2026-key"; the key-sensitivity test changes one bit of its last character. Salts are random in normal use; in the experiments they are derived deterministically from the file and cover names so that every table can be reproduced exactly. The original method (Shukur *et al*., 2018) [31] was run at the same payloads with the same number of LSBs per channel, using its own position generator.

4.2. Encryption Quality

Table 2 reports, for each file, the Shannon entropy of the plaintext, of the original ciphertext and of the proposed ciphertext; the correlation between adjacent ciphertext bytes; and NPCR/UACI (Wu *et al*., 2011) [37] between two ciphertexts when one plaintext bit is flipped, when one password bit is flipped and when one bit of the salt is flipped. The proposed ciphertext reaches 7.9795-7.9990 bits/byte for all file types, including low-entropy text (5.4359 bits/byte plaintext) and executables (6.3093), while the original substitution leaves visible structure (7.3994-7.9979). Fig. 4 illustrates the byte histograms for the TXT and EXE files. Adjacent-byte correlation is at most 0.0140 in absolute value (Fig. 5). The decisive difference is diffusion: the original cipher changes at most 0.02% of bytes when one plaintext bit changes, whereas the proposed cipher changes 99.54-99.68% with UACI of 33.36-33.94%, close to the ideal values of 99.61% and 33.46% for random bytes. A different salt with the same file and password changes 99.54-99.66% of the bytes, so repeated messages under one password are unrelated. Every file was decrypted exactly.

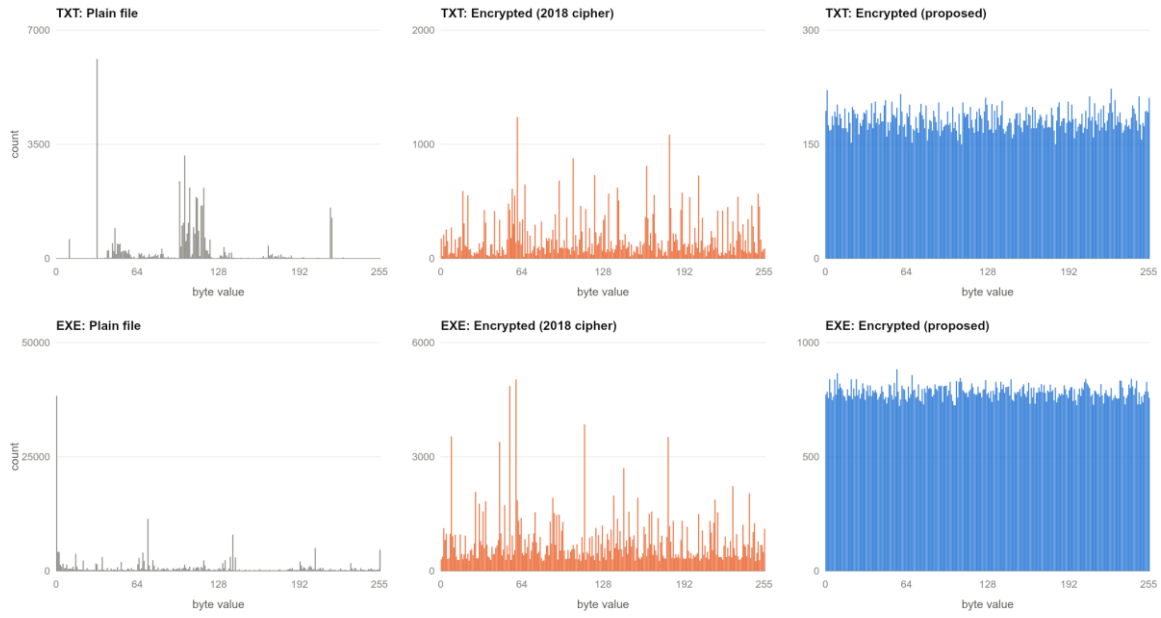


Fig 4: Byte histograms of the plain file, the 2018 ciphertext and the proposed ciphertext (TXT top, EXE bottom)

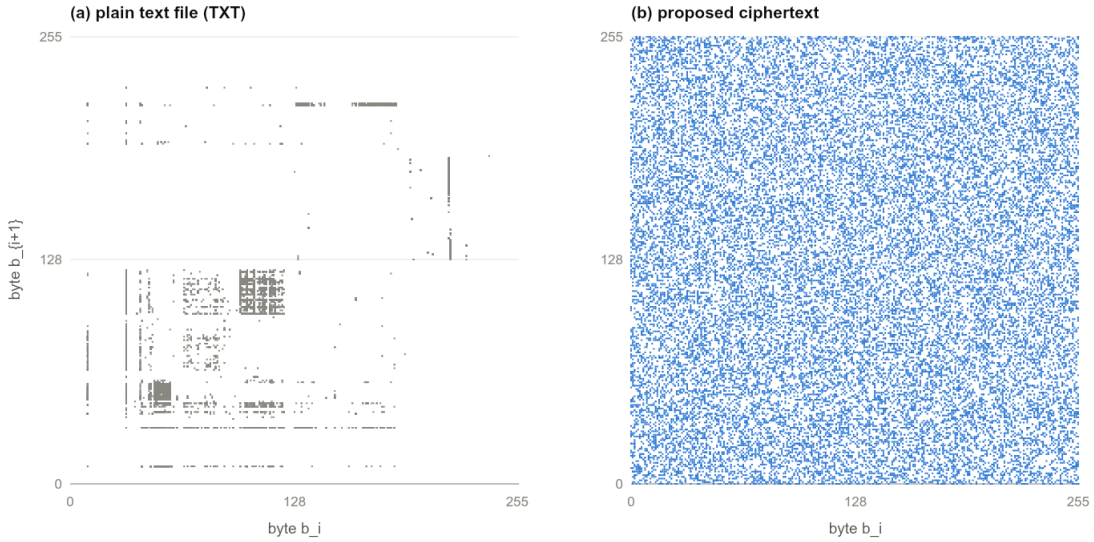


Fig 5: Pairs of adjacent bytes (b_i, b_i+1) of the TXT file before and after encryption

4.3. Imperceptibility

With one bit per sample, a random payload bit already equals the LSB with probability 1/2, so on average one sample in two is changed by exactly ±1, both for LSB matching and for LSB replacement. For a colour image with three samples per pixel the expected mean squared error is therefore bpp/6, and the PSNR is fixed by the payload alone:

$$\mathrm{PSNR}_{\mathrm{ref}}(\mathrm{bpp}) = 10\log_{10}\left(255^2 \cdot 6 / \mathrm{bpp}\right)\ \mathrm{dB}\ (\text{colour}),$$
$$10\log_{10}\left(255^2 \cdot 2 / \mathrm{bpp}\right)\ \mathrm{dB}\ (\text{grayscale})$$

Table 3 reports PSNR, SSIM (computed globally over the image) and NCC for each USC-SIPI cover at 0.5, 1, 2 and 3 bpp. Depths 1, 1, 1 and 2 were selected automatically. At k = 1 the three schemes have essentially the same PSNR (mean 58.92, 55.92 and 52.90 dB at 0.5, 1 and 2 bpp, against 58.92, 55.91 and 52.90 dB from the formula). At 3 bpp (k = 2), OPAP raises the mean PSNR from 47.12 dB for plain replacement and 47.12 dB for the original method to 49.32 dB. SSIM is at least 0.9998 in all cases.

Table 3: Imperceptibility on the USC-SIPI covers against payload (512×512)

Image	bpp	Payload (B)	k	PSNR proposed	PSNR replacement	PSNR 2018	SSIM	NCC	Extraction
Airplane	0.5	16348	1	58.947	58.933	58.909	0.999979	1.000000	exact
Airplane	1	32732	1	55.911	55.908	55.909	0.999958	1.000000	exact
Airplane	2	65500	1	52.911	52.899	52.903	0.999915	1.000003	exact
Airplane	3	98268	2	49.369	47.127	47.126	0.999809	1.000016	exact
Baboon	0.5	16348	1	58.916	58.917	58.915	0.999987	0.999999	exact
Baboon	1	32732	1	55.918	55.908	55.910	0.999974	1.000000	exact
Baboon	2	65500	1	52.897	52.890	52.906	0.999948	0.999998	exact
Baboon	3	98268	2	49.388	47.160	47.155	0.999883	0.999996	exact
Peppers	0.5	16348	1	58.901	58.917	58.940	0.999991	0.999995	exact
Peppers	1	32732	1	55.921	55.903	55.909	0.999981	0.999997	exact
Peppers	2	65500	1	52.895	52.914	52.907	0.999962	1.000002	exact
Peppers	3	98268	2	49.161	47.071	47.064	0.999911	1.000012	exact
Sailboat	0.5	16348	1	58.928	58.913	58.913	0.999991	0.999999	exact
Sailboat	1	32732	1	55.913	55.908	55.934	0.999982	1.000004	exact
Sailboat	2	65500	1	52.906	52.901	52.905	0.999964	0.999991	exact
Sailboat	3	98268	2	49.367	47.124	47.117	0.999919	1.000019	exact

Table 4 extends the measurement to the 1,800 DIV2K covers and reports, besides the global SSIM, the standard windowed SSIM of Wang *et al.* (2004) [34] (11×11 Gaussian window, σ = 1.5), which is more sensitive to local changes. The mean PSNR of the proposed scheme is 65.91, 62.90, 59.89, 55.91 dB at 0.1, 0.2, 0.4, 1 bpp, within 0.002 dB of the formula, and its standard deviation over all covers is at most 0.027 dB: at these payloads PSNR depends on the payload and not on the image. The windowed SSIM varies more (Fig. 7) and is lowest at 1 bpp (mean 0.998895); LSB replacement and the original method give the same values to within the reported precision, as expected from the equal number of ±1 changes.

Table 4: Imperceptibility on 1,800 DIV2K covers (mean ± standard deviation)

Scheme	bpp	PSNR (dB)	Reference PSNR (dB)	SSIM global	SSIM windowed	NCC
Proposed (matching)	0.1	65.914 ± 0.027	65.912	0.999998 ± 0.000001	0.999891 ± 0.000048	0.999999
Proposed (matching)	0.2	62.902 ± 0.020	62.902	0.999995 ± 0.000003	0.999782 ± 0.000096	0.999998
Proposed (matching)	0.4	59.891 ± 0.013	59.892	0.999990 ± 0.000006	0.999562 ± 0.000194	0.999995
Proposed (matching)	1	55.912 ± 0.009	55.912	0.999975 ± 0.000014	0.998895 ± 0.000512	0.999989
LSB replacement	0.1	65.914 ± 0.027	65.912	0.999998 ± 0.000001	0.999891 ± 0.000048	0.999999
LSB replacement	0.2	62.902 ± 0.019	62.902	0.999995 ± 0.000003	0.999782 ± 0.000096	0.999999
LSB replacement	0.4	59.892 ± 0.014	59.892	0.999990 ± 0.000006	0.999563 ± 0.000193	0.999997
LSB replacement	1	55.913 ± 0.008	55.912	0.999975 ± 0.000014	0.998901 ± 0.000509	0.999993
Shukur *et al.* (2018) [31]	0.1	65.914 ± 0.027	65.912	0.999998 ± 0.000001	0.999891 ± 0.000048	0.999999
Shukur *et al.* (2018) [31]	0.2	62.903 ± 0.019	62.902	0.999995 ± 0.000003	0.999782 ± 0.000096	0.999998
Shukur *et al.* (2018) [31]	0.4	59.892 ± 0.014	59.892	0.999990 ± 0.000006	0.999563 ± 0.000193	0.999997
Shukur *et al.* (2018) [31]	1	55.912 ± 0.008	55.912	0.999975 ± 0.000014	0.998901 ± 0.000508	0.999993

Fig. 6 places all measured points on the reference curves. Both cover sets fall on the colour curve up to 2 bpp; the 3 bpp point, at k = 2, lies below it. Fig. 8 shows Peppers at 3 bpp with the amplified difference image, in which the changes are spread uniformly over the image.

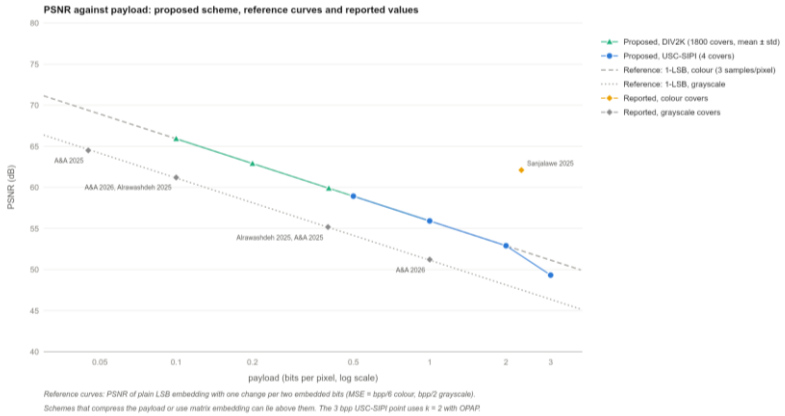


Fig 6: PSNR against payload: proposed scheme on both cover sets, reference curves of 1-LSB embedding, and values reported by recent schemes

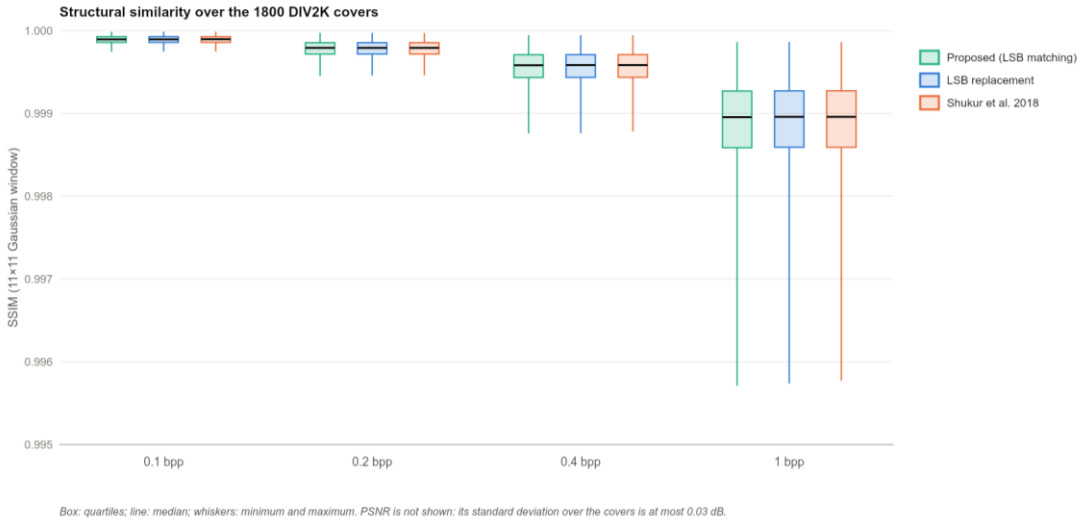


Fig 7: Distribution of the windowed SSIM over the 1,800 DIV2K covers

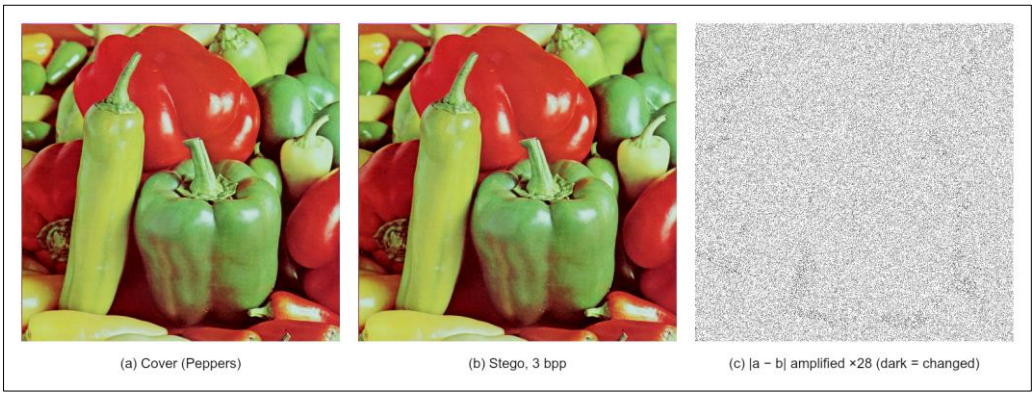


Fig 8: (a) Peppers cover, (b) stego image at 3 bpp, (c) amplified absolute difference

4.4. Hiding Real Files

Table 5 lists the result of hiding each of the eleven files in Peppers with automatic depth selection. Files up to about 89 KB are embedded at k = 1 with PSNR above 51.5 dB; the PDF (124.8 KB) requires k = 2 (48.026 dB) and the EXE (196.0 KB, 6.126 bpp) requires k = 3 (42.033 dB). Every file was extracted bit-exactly, with the correct extension, from the stego image alone and the password.

Table 5: Hiding eleven file types in Peppers (512×512); times include PNG decoding and encoding

File	Size (KB)	bpp	k	PSNR (dB)	SSIM	NCC	Hide (ms)	Extract (ms)	Exact
ZIP	88.8	2.777	1	51.479	0.999948	0.999999	145	102	yes
WAV	62.5	1.956	1	52.986	0.999963	0.999995	932	87	yes
RTF	23.2	0.725	1	57.305	0.999986	0.999999	104	71	yes
PNG	58.8	1.837	1	53.276	0.999965	1.000000	129	85	yes
BMP	75.1	2.346	1	52.213	0.999956	0.999993	306	105	yes
EXE	196.0	6.126	3	42.033	0.999538	0.999999	185	151	yes
DOCX	19.1	0.598	1	58.154	0.999989	1.000001	595	69	yes
PDF	124.8	3.900	2	48.026	0.999884	1.000026	141	844	yes
JPG	9.3	0.291	1	61.271	0.999994	1.000002	95	72	yes
TXT	45.5	1.423	1	54.372	0.999973	0.999999	118	380	yes
XLSX	19.0	0.594	1	58.171	0.999989	0.999996	108	67	yes

4.5. Execution Time

The cost of the cipher is dominated by the two counter streams that protect every chaotic output. Table 6 and Fig. 9 compare the two primitives on the same 722 KB of plaintext (the eleven files): with SHA-256 counter blocks the keystream is produced at 1.01 MB/s, and with AES-256-CTR at 5.83 MB/s (×5.8); complete encryption rises from 0.47 to 2.39 MB/s (×5.1). The chaotic map alone runs at about 10.2 MB/s, so with AES the map, not the hedge, becomes the main cost. Per file, encryption and decryption take 4-79 ms (Table 2); building the full 262,144-pixel permutation and embedding take 51-85 ms per image; and the complete hide and extract operations, including PNG input and output, take 95-932 ms and 67-844 ms (Table 5).

Table 6: Throughput of the counter-stream primitives (single thread)

Primitive	Keystream (MB/s)	Chaotic map only (MB/s)	Encryption (MB/s)	Encryption of 11 files (ms)
SHA-256 counter blocks	1.006	11.050	0.472	1493
AES-256-CTR	5.829	10.169	2.391	295

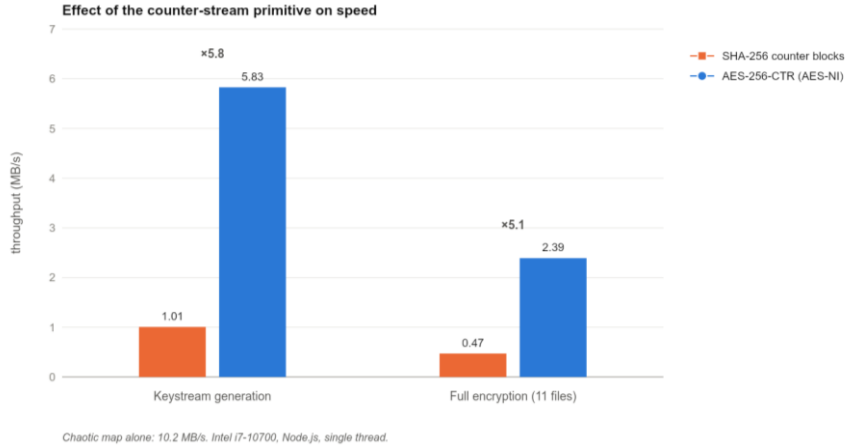


Fig 9: Effect of the counter-stream primitive on keystream and encryption throughput

4.6. Comparison with Related Work

Table 1 places the proposed scheme among the reviewed works. Direct PSNR comparison is meaningful only at equal payload and cover type, and Fig. 6 makes this explicit: where the reviewed papers report a payload that can be converted to bpp, their PSNR values fall on or slightly below the grayscale reference curve (Aljughaiman and Alrawashdeh, 2025, 2026; Alrawashdeh *et al*., 2025) [2, 3, 4], exactly as the proposed scheme falls on the colour curve. A grayscale image has one sample per pixel instead of three, so the same bpp costs about 4.8 dB more; this, and not the embedding method, accounts for most of the difference between those schemes and ours. The only reported point above its curve (Sanjalawe *et al*., 2025) [29] is consistent with compressing the payload with Huffman coding before embedding, which reduces the number of embedded bits per character. The PSNR values of 70-88 dB reported for text payloads of 2-14 KB (Rahman *et al*., 2025; Banoori *et al*., 2025) [8, 27] cannot be placed on Fig. 6 without the exact cover size and PSNR definition: in a 512×512 colour cover such payloads correspond to 0.06-0.44 bpp, where the reference curve gives 59.5-68.0 dB, so the reported values exceed what an embedding with one ±1 change per two bits can reach by 10 dB or more. We therefore do not compare with them numerically. The distinguishing properties of the proposed scheme are therefore not a higher PSNR but the combination of arbitrary files with blind, integrity-checked recovery, per-message salted keys, colour covers, a standard randomness test suite and an evaluation on 1,800 covers.

5. Security Analysis

5.1. Key Space

All key material is derived from K = SHA-256(P), so the key space is 2^256, far above the 2^100 threshold usually required (Alvarez and Li, 2006) [5]. The effective key space, however, is bounded by the entropy of the password itself; a brute-force attacker would enumerate passwords rather than 256-bit keys. The salt does not add secret entropy, but it prevents an attacker from precomputing the sub-keys of a password list once for all messages. Users should choose long random passphrases, and a memory-hard key-derivation function (such as Argon2) can replace the single SHA-256 without changing any other part of the scheme.

5.2. Key and Salt Sensitivity

Flipping one bit of the last password character changes 99.60-99.67% of the ciphertext bytes with UACI of 33.20-33.78% (Table 2), i.e. the two ciphertexts are statistically unrelated (Fig. 10); flipping one bit of the salt has the same effect (99.54-99.66%). Extraction with the modified password fails at the header check (Table 11), because the permutation, the header mask and the diffusion keystreams all change completely.

5.3. Steganalysis

Three classical detectors were applied to every USC-SIPI stego image: the chi-square attack (Westfeld and Pfitzmann, 2000) [35], RS analysis with mask [0, 1, 1, 0] (Fridrich *et al*., 2001) [17] and sample-pair analysis (SPA; Dumitrescu *et al*., 2003) [13]. RS and SPA estimate the embedding rate; Table 7 reports the estimate on the stego image minus the estimate on the clean cover, so that a value near zero means no detection. The results separate clearly by embedding rule. For LSB replacement, both in the proposed ablation and in the original method, RS detects embedding in every case, with estimates of 0.11-0.70 and 0.13-0.70 respectively; they grow with payload up to 2 bpp and fall at 3 bpp, where the depth-2 changes are spread over fewer pixels. For the proposed LSB-matching/OPAP embedding on Airplane, Baboon and Sailboat, the RS estimate never exceeds the clean-cover level by more than 0.054, and SPA stays below 0.009. The chi-square attack returned a probability of embedding close to zero for all random-order embeddings except Baboon at 2 bpp with replacement (0.8445) and the original method on Baboon at 2 and 3 bpp (0.9957, 0.5908). Fig. 11 plots the mean RS and SPA estimates.

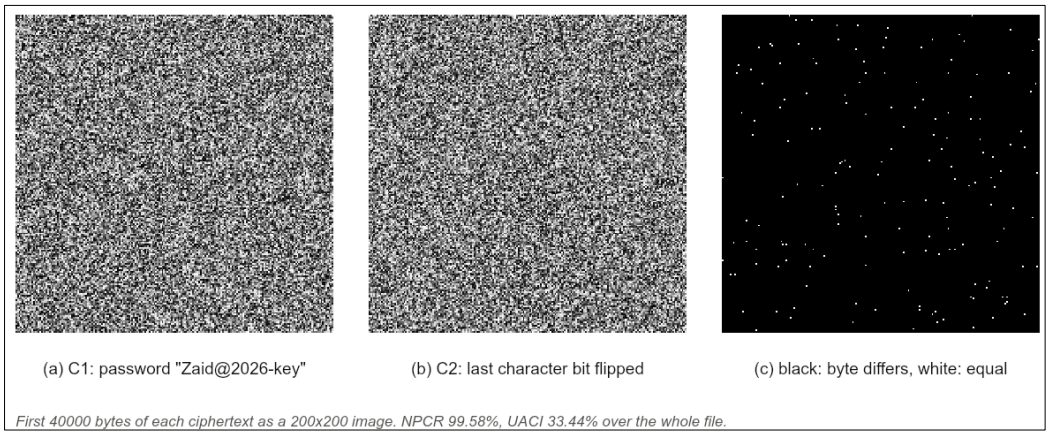


Fig 10: Key sensitivity: ciphertexts of the same file under two passwords that differ in one bit, and the positions where they agree

Fig. 12 and Fig. 13 show why. RS analysis relies on the four group counts R_M, R_−M, S_M and S_−M being pairwise equal in a clean image; LSB replacement drives each pair apart in proportion to the embedding rate, whereas ±1 matching leaves them together at every rate. In the histogram, replacement equalises the counts of each pair of values (2i, 2i+1), the artefact on which the chi-square attack and SPA are built; matching leaves the shape of the histogram intact.

Peppers behaves differently: its clean cover already yields RS = 0.091 and SPA = 0.037, a known bias of these estimators on some natural images, and with LSB matching the RS estimate rises by 0.026-0.099 above that level. This rise is much smaller than that caused by replacement at the same payload, but it is not negligible. Our acceptance criterion, an RS increase of at most 0.03 over the clean cover at 1 bpp, was therefore met on Airplane, Baboon and Sailboat and missed

on Peppers (0.042), where the non-zero clean-cover estimate makes RS unreliable; we do not claim undetectability on this image. Two limits of this analysis must be stated precisely. First, LSB matching defeats the structural assumption of the chi-square, RS and SPA detectors at k = 1; it is not undetectable in general, and no resistance is claimed against detectors designed for ±1 embedding, such as the histogram-characteristic-function centre of mass (Harmsen and Pearlman, 2003; Ker, 2005) [18, 22], or against rich-model and deep-learning steganalysers (Fridrich and Kodovsky, 2012; Boroumand *et al*., 2019) [9, 16]. Second, at k ≥ 2 the embedded LSB plane is still replaced. The low RS and SPA values measured at 3 bpp (k = 2) are a side effect of OPAP: its ±4 correction moves a value out of its original LSB pair {2i, 2i+1}, which invalidates the within-pair flipping model on which RS and SPA are built. This was observed at k = 2 only and is not generalised; OPAP was designed to reduce distortion, not to resist steganalysis.

Table 7: Steganalysis: RS and SPA estimated embedding rate minus the clean-cover estimate

Image	bpp	RS proposed	RS replacement	RS 2018	SPA proposed	SPA replacement	SPA 2018
Airplane	0.5	0.0011	0.1543	0.1710	0.0000	0.0423	0.0431
Airplane	1	0.0073	0.3257	0.3473	0.0000	0.1346	0.1380
Airplane	2	0.0056	0.7007	0.6964	0.0000	0.0000	0.0000
Airplane	3	0.0000	0.4765	0.4977	0.0000	0.2149	0.2201
Baboon	0.5	0.0000	0.1078	0.1365	0.0000	0.0000	0.0000
Baboon	1	0.0161	0.2660	0.3346	0.0000	0.0000	0.0000
Baboon	2	0.0094	0.6552	0.6743	0.0000	0.1859	0.0642
Baboon	3	0.0186	0.4307	0.5069	0.0000	0.0373	0.0310
Peppers	0.5	0.0258	0.1507	0.1728	0.0265	0.0811	0.0819
Peppers	1	0.0417	0.3020	0.3225	0.0367	0.1725	0.1762
Peppers	2	0.0794	0.6295	0.6350	0.0759	-0.0370	-0.0370
Peppers	3	0.0993	0.4844	0.5160	0.0558	0.0829	-0.0370
Sailboat	0.5	0.0044	0.1486	0.1327	0.0018	0.0676	0.0704
Sailboat	1	0.0163	0.3136	0.3035	0.0087	0.1759	0.1608
Sailboat	2	0.0537	0.6644	0.6337	0.0016	-0.0006	-0.0006
Sailboat	3	0.0088	0.4550	0.4511	0.0031	0.0844	0.0903

Highlighted rows: Peppers, whose clean cover is already biased (see text).

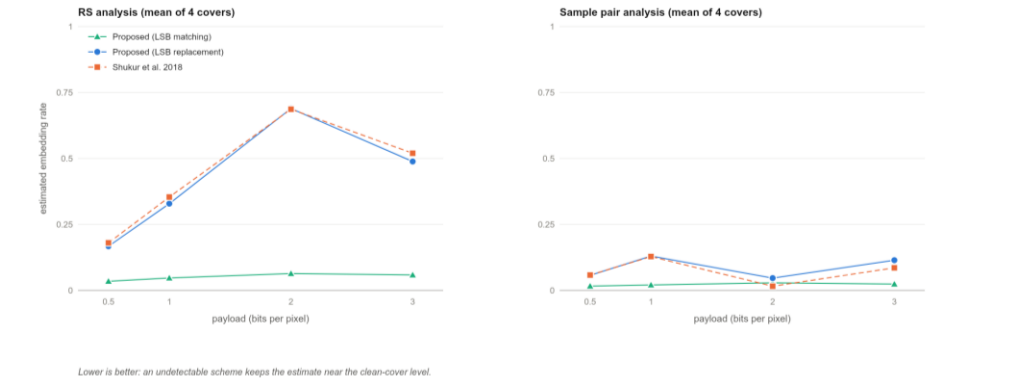


Fig 11: Mean RS and SPA estimates of the four USC-SIPI covers against payload

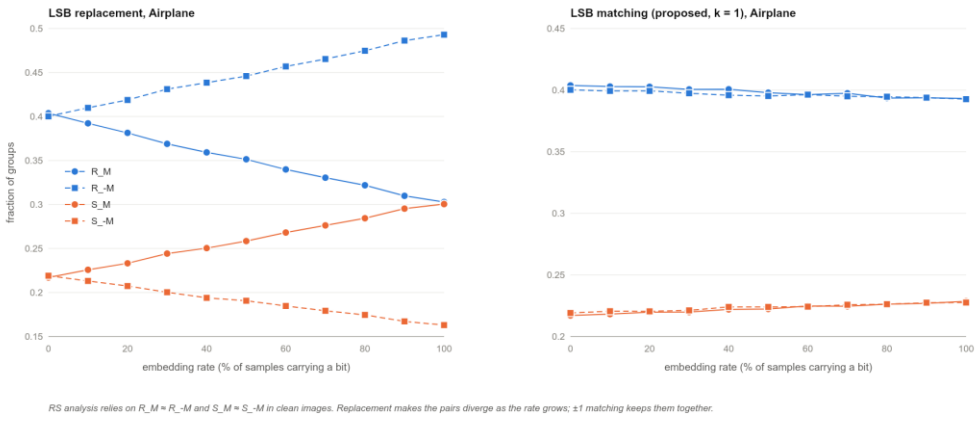


Fig 12: RS diagram of Airplane: group counts against embedding rate for LSB replacement (left) and the proposed LSB matching (right)

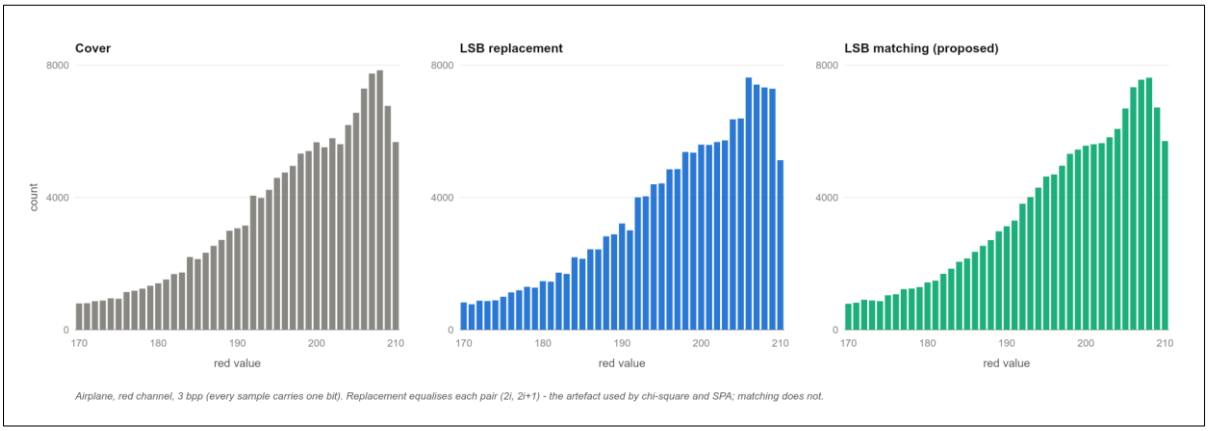


Fig 13: Red-channel histogram of Airplane at 3 bpp: cover, LSB replacement and LSB matching

5.4. Degradation of the Chaotic Map

To quantify dynamical degradation, the generator was run in three configurations on the same keys: the plain 2D-LSCM, the map with the anti-degradation perturbation only, and the full hedged generator used by the scheme. The key set contained 18 keys known to collapse under the plain map and 10,000 random salted sub-keys. A key was counted as cycling if fewer than 99% of its outputs were distinct; 20,000 outputs per key were used for the plain map and 100,000 for the other two. The keystream bytes were pooled and tested with a chi-square test on 256 bins. Table 8 shows that the plain map cycles for 8.4% of random keys and produces a grossly non-uniform byte distribution, whereas the perturbation alone removes every cycle, including those of the known collapsing keys, and both protected configurations pass the uniformity test at the 1% level. Fig. 14 shows a collapsed key: the plain map visits only three values, while the perturbed map fills the return map and its number of distinct outputs grows linearly with the number of iterations. The permutation under the experiment key now selects between 99.0% and 101.3% of the expected number of pixels in every band of 32 rows at 50% load (maximum relative deviation 1.3%), which is consistent with a uniform random permutation.

Table 8: Degradation test of the 2D-LSCM generator

Configuration	Keys	Outputs per key	Keys with cycles	Known collapsing keys still cycling	Min. distinct ratio	χ^2 (255 df)	Uniform ($p > 0.01$)
Plain 2D-LSCM	10018	20000	856	18 of 18	0.00005	21918.67	no
Perturbed	10018	100000	0	0 of 18	0.99892	252.47	yes
Perturbed + hedge (proposed)	10018	100000	0	0 of 18	0.99993	295.94	yes

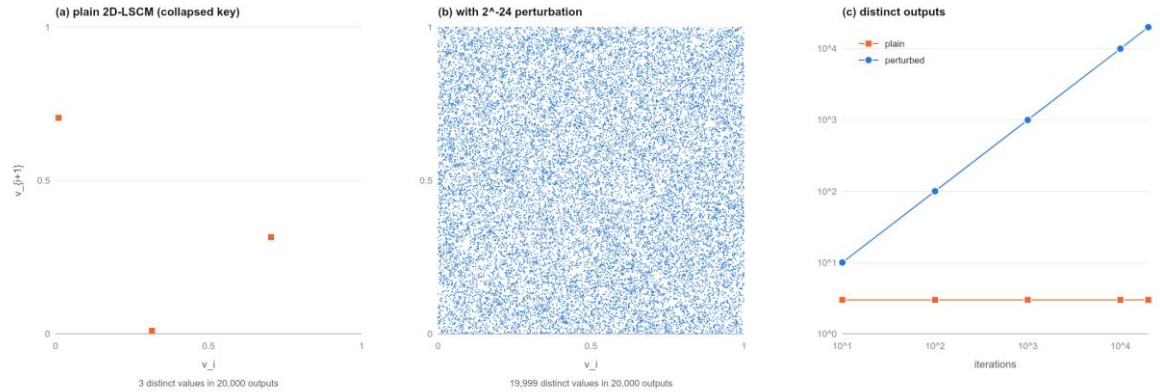


Fig 14: Return map of a collapsing key under the plain 2D-LSCM (a) and with the 2^-24 perturbation (b); number of distinct outputs against iterations (c)

5.5. Statistical Randomness (NIST SP 800-22)

The absence of short cycles does not by itself make a sequence random. The NIST SP 800-22 suite (Rukhin *et al*., 2010) [28] was therefore applied to three sources, each as 100 sequences of 2^20 bits under different keys: (a) the perturbed 2D-LSCM without the hedge, (b) the full generator used by the scheme and (c) the ciphertext of an all-zero 128 KB file under 100 different salts, which isolates the cipher from any structure in the plaintext. Each test implementation was first verified against the worked examples of the NIST document; the discrete Fourier transform test could not be verified, because the worked example in the document is not reproducible from its own intermediate values, and it is excluded rather than reported unverified. The remaining 11 statistics (Frequency, BlockFrequency, Runs, LongestRun, Rank, NonOverlappingTemplate, Serial-1, Serial-2, ApproximateEntropy, CumulativeSums-fwd, CumulativeSums-rev) were evaluated at $\alpha = 0.01$; a source

passes a statistic when the proportion of passing sequences is at least 0.9602 (97 of 100 sequences) and the P-values are uniform ($P \geq 0.0001$). Table 9 and Fig. 15 show that sources (b) and (c) pass all 11 statistics, whereas the perturbed map alone fails Serial-1 (proportion 0.96) and ApproximateEntropy (proportion 0.95). The perturbation cures the cycles, but the map's output still carries short-range structure that the serial and approximate-entropy tests detect; the AES-CTR hedge is what makes the generator pass, which is why the scheme uses both.

Table 9: NIST SP 800-22: proportion of 100 sequences passing (uniformity P-value in brackets)

Statistic	(a) perturbed 2D-LSCM, no hedge	(b) full generator (scheme)	(c) ciphertext of all-zero file
Frequency	0.99 [0.4012]	0.98 [0.1154]	1 [0.0329]
BlockFrequency	0.98 [0.4750]	0.98 [0.4559]	1 [0.1719]
Runs	0.98 [0.2622]	0.99 [0.7598]	1 [0.3345]
LongestRun	0.98 [0.8978]	1 [0.6993]	1 [0.1719]
Rank	1 [0.7792]	1 [0.0192]	0.97 [0.6371]
NonOverlappingTemplate	1 [0.1154]	0.99 [0.3669]	1 [0.0554]
Serial-1	0.96 [0.6163] FAIL	0.99 [0.6579]	0.99 [0.2622]
Serial-2	0.99 [0.2133]	1 [0.8343]	0.99 [0.6993]
ApproximateEntropy	0.95 [0.4750] FAIL	0.99 [0.1816]	1 [0.7399]
CumulativeSums-fwd	0.99 [0.0192]	0.98 [0.1373]	1 [0.0006]
CumulativeSums-rev	0.99 [0.2622]	0.99 [0.6163]	0.99 [0.0329]

5.6. Chosen-Plaintext Resistance

A substitution cipher with a fixed table can be broken with chosen plaintexts; the diffusion layer is meant to prevent this. Table 10 gives the results of encrypting constant 1 MB files under the same password. The ciphertexts of an all-zero and an all-0xFF file have entropy 7.99986 and 7.99983 bits/byte, and differ from each other in 99.61% of their bytes. Changing a single byte of either file at its beginning, middle or end changes about 99.6% of the ciphertext; a change of the last byte changes every byte, because the backward pass propagates it through the whole sequence by a bijection. The attacker therefore learns nothing about the table from differences of chosen plaintexts.

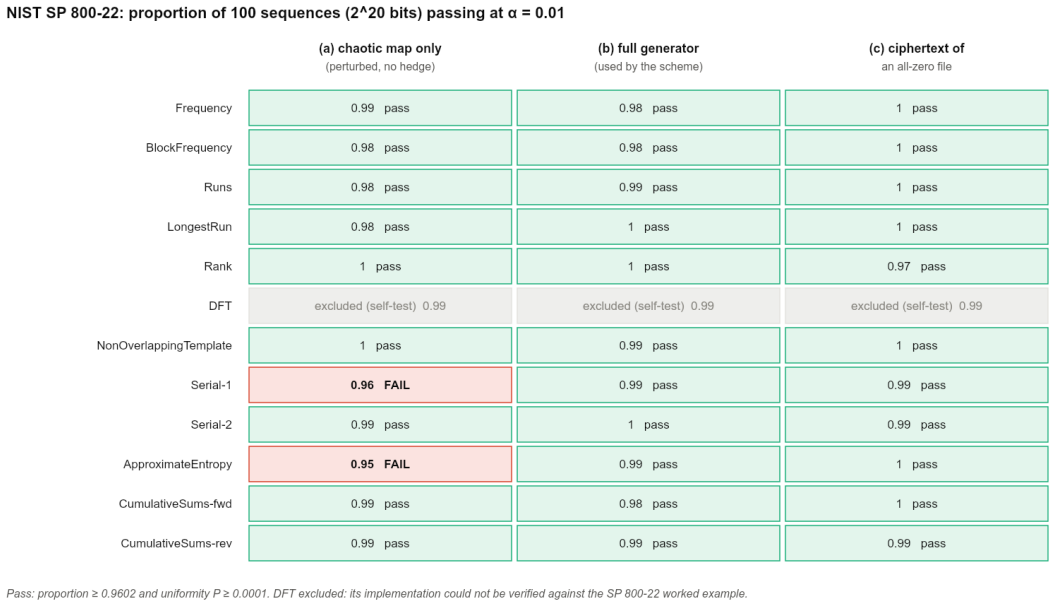


Fig 15: NIST SP 800-22 results of the three sources

Table 10: Chosen-plaintext tests on constant files (NPCR/UACI in %)

Plaintext	Entropy (bits/byte)	Change first byte	Change middle byte	Change last byte
all-zero	7.999861	99.61/33.47	99.62/33.45	100.00/33.63
all-0xFF	7.999827	99.61/33.46	99.62/33.46	100.00/33.61

5.7. Coverage of the Original Position Generator

The maximum fraction of pixels that the modular generator of Shukur *et al.* (2018) [31] can address before it stops producing new positions was found by bisection on the number of requested pixels. On a 256×256 image it reaches at least 99.9% of the pixels, but on a 512×512 image it stalls between 85.74% and 85.84%, so payloads that need more pixels never finish embedding. Fig. 16 shows the coverage against the number of generator attempts: after 10^8 attempts it saturates at 99.9% and 85.8% of the pixels respectively. This is why the original method was compared only at payloads needing at most 80% of the pixels, and it is a documented limitation of that method. The Fisher-Yates permutation removes the problem entirely (Table 11).

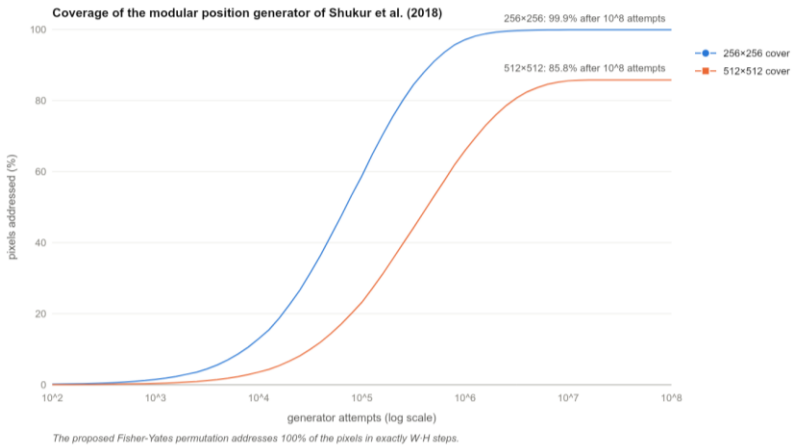


Fig 16: Fraction of pixels addressed by the modular generator of Shukur *et al*. (2018) against the number of attempts

5.8. Limitations

- **Fragility.** LSB embedding is fragile by design: any lossy compression, resizing or filtering of the stego image destroys the payload. The CRC-32 makes such damage detectable but not correctable. The stego image must be stored and transmitted as PNG or another lossless format.
- **Scope of steganalysis.** The advantage of the ±1 embedding is established only against the chi-square, RS and sample-pair attacks; it is not claimed against detectors targeted at LSB matching, rich-model or learned steganalysers, and one of the four covers shows a small RS rise at k = 1.
- **Integrity versus authenticity.** The CRC-32 detects accidental and naive modification but is not a cryptographic authenticator; an HMAC under a further sub-key would provide authenticity.
- **Shared permutation.** The pixel permutation depends only on the password, because the receiver needs it to find the salt. Two stego images made with the same password therefore use the same pixel order; an attacker who also holds both covers can see that the changed pixels follow the same order, although the salt keeps the embedded bits unrelated. A separate password per correspondent limits this exposure.
- **Password strength.** The security of the key depends on the password; a memory-hard key-derivation function is recommended for deployment.
- **High payloads.** Payloads above 3 bpp (one bit per colour channel) require $k \geq 2$, where the LSB plane is replaced and the observed resistance to RS and SPA rests on a side effect of OPAP that was verified only at k = 2.

6. Conclusion

This study examined two components on which chaos-based crypto-steganographic schemes commonly rely, and found that each can fail silently. A modular pixel-position generator could not address more than 85.7% of a 512×512 image, and the unperturbed 2D-LSCM fell into short cycles in double precision for 8.4% of random keys. A 2^-24 perturbation removed the cycles, but only its combination with an AES-256-CTR hedge satisfied the NIST SP 800-22 criteria.

An improved, blindly recoverable scheme was designed using password-derived SHA-256 sub-keys and per-message salting. The integration of multi-level substitution, two-pass non-linear diffusion and a Fisher–Yates permutation achieved full use of the cover image. Across eleven file types the ciphertext entropy reached 7.9795-7.9990 bits/byte, a one-bit change of the plaintext, password or salt altered 99.54-99.68% of the ciphertext bytes, and the cipher withstood the chosen-plaintext tests. Encrypted headers and per-message salts enabled blind recovery of the payload, its file type and its integrity. On 1,800 DIV2K cover images the stego PSNR followed the theoretical curve of ±1 embedding, from 65.9 dB at 0.1 bpp to 55.9 dB at 1 bpp, and the same curves account for most of the PSNR differences reported between recent schemes. Chi-square, RS and sample-pair analysis detected LSB replacement in every case but did not detect the proposed embedding on three of the four USC-SIPI covers.

Future research will explore content-adaptive position selection to mitigate detection by modern machine-learning steganalysis. Additionally, incorporating memory-hard key derivation functions and HMAC structures will further fortify defense against targeted attacks.

References

1. Agustsson E, Timofte R. NTIRE 2017 challenge on single image super-resolution: Dataset and study. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 2017:126-135. doi:10.1109/CVPRW.2017.150
2. Aljughaiman A, Alrawashdeh R. Content-adaptive LSB steganography with saliency fusion, ACO dispersion, and hybrid encryption with ablation study. Scientific Reports; 2025. doi:10.1038/s41598-025-33920-9
3. Aljughaiman A, Alrawashdeh R. Optimization-driven steganographic system based on fused maps and blowfish encryption. Scientific Reports; 2026. doi:10.1038/s41598-026-35556-9
4. Alrawashdeh R, Almuhammadi S, Niazi M, Mahmud M, Rahman MM. Secure edge-guided adaptive image steganography using HED-based attention maps and CNN. Scientific Reports; 2025. doi:10.1038/s41598-025-27150-2
5. Alvarez G, Li S. Some basic cryptographic requirements for chaos-based cryptosystems. International Journal of Bifurcation and Chaos. 2006;16(8):2129-2151.

doi:10.1142/S0218127406015970
6. Bailey K, Curran K. An evaluation of image based steganography methods. Multimedia Tools and Applications. 2006;30:55-88.
7. Baluja S. Hiding images in plain sight: Deep steganography. Advances in Neural Information Processing Systems (NeurIPS). 2017;30.
8. Banoori SZ, Khan W, Rahman S, Masood F, Salam A, Amin F, *et al*. An improved hybrid image steganography method using AES algorithm. Scientific Reports; 2025. doi:10.1038/s41598-025-28140-0
9. Boroumand M, Chen M, Fridrich J. Deep residual network for steganalysis of digital images. IEEE Transactions on Information Forensics and Security. 2019;14(5):1181-1193. doi:10.1109/TIFS.2018.2871749
10. Chan CK, Cheng LM. Hiding data in images by simple LSB substitution. Pattern Recognition. 2004;37(3):469-474. doi:10.1016/j.patcog.2003.08.007
11. Cheddad A, Condell J, Curran K, Mc Kevitt P. Digital image steganography: Survey and analysis of current methods. Signal Processing. 2010;90(3):727-752. doi:10.1016/j.sigpro.2009.08.010
12. Dsa KS, Patro KAK. Enhancing image security: a lightweight crypto-steganographic approach for optimal quality. Multimedia Tools and Applications; 2026. doi:10.1007/s11042-026-21459-4
13. Dumitrescu S, Wu X, Wang Z. Detection of LSB steganography via sample pair analysis. IEEE Transactions on Signal Processing. 2003;51(7):1995-2007. doi:10.1109/TSP.2003.812753
14. Durstenfeld R. Algorithm 235: Random permutation. Communications of the ACM. 1964;7(7):420. doi:10.1145/364520.364540
15. Dworkin M. Recommendation for block cipher modes of operation: Methods and techniques. NIST Special Publication. 2001:800-38A. doi:10.6028/NIST.SP.800-38A
16. Fridrich J, Kodovsky J. Rich models for steganalysis of digital images. IEEE Transactions on Information Forensics and Security. 2012;7(3):868-882. doi:10.1109/TIFS.2012.2190402
17. Fridrich J, Goljan M, Du R. Reliable detection of LSB steganography in color and grayscale images. Proceedings of the ACM Workshop on Multimedia and Security, pp. 2001:27-30. doi:10.1145/1232454.1232466
18. Harmsen JJ, Pearlman WA. Steganalysis of additive-noise modelable information hiding. Proceedings of SPIE, Security and Watermarking of Multimedia Contents V. 2003;5020:131-142. doi:10.1117/12.476813
19. Holub V, Fridrich J, Denemark T. Universal distortion function for steganography in an arbitrary domain. EURASIP Journal on Information Security. 2014;2014:1. doi:10.1186/1687-417X-2014-1
20. Hua Z, Jin F, Xu B, Huang H. 2D Logistic-Sine-coupling map for image encryption. Signal Processing. 2018;149:148-161. doi:10.1016/j.sigpro.2018.03.010
21. Kadhim IJ, Premaratne P, Vial PJ, Halloran B. Comprehensive survey of image steganography: Techniques, evaluations, and trends in future research. Neurocomputing. 2019;335:299-326. doi:10.1016/j.neucom.2018.06.075
22. Ker AD. Steganalysis of LSB matching in grayscale images. IEEE Signal Processing Letters. 2005;12(6):441-444. doi:10.1109/LSP.2005.847889
23. Khalil N, Sarhan A, Alshewimy MAM. A secure image steganography based on LSB technique and 2D chaotic maps. Computers & Electrical Engineering. 2024;119:109566. doi:10.1016/j.compeleceng.2024.109566
24. Kumar R, Singhal S, Sharma VK. Efficient image steganography method using contourlet transform and geometric-based pixel encryption for enhanced security. Scientific Reports; 2026. doi:10.1038/s41598-026-41168-0
25. Li S, Chen G, Mou X. On the dynamical degradation of digital piecewise linear chaotic maps. International Journal of Bifurcation and Chaos. 2005;15(10):3119-3151. doi:10.1142/S0218127405014052
26. Naveen R, Kumar AP, Roshan S. A hybrid blowfish-based cryptography and chaotic quantization steganography framework with genetic key generation. Scientific Reports; 2026. doi:10.1038/s41598-026-47129-x
27. Rahman S, Uddin J, Hussain H, Shah S, Salam A, Amin F, *et al*. A novel and efficient digital image steganography technique using least significant bit substitution. Scientific Reports; 2025. doi:10.1038/s41598-024-83147-3
28. Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, *et al*. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22 Rev. 1a; 2010. doi:10.6028/NIST.SP.800-22r1a
29. Sanjalawe Y, Al-E'mari S, Fraihat S, Abualhaj M, Alzubi E. A deep learning-driven multi-layered steganographic approach for enhanced data security. Scientific Reports; 2025. doi:10.1038/s41598-025-89189-5
30. Sharp T. An implementation of key-based digital signal steganography. Information Hiding (IH, Lecture Notes in Computer Science, 2137: 13-26; 2001. doi:10.1007/3-540-45496-9_2
31. Shukur BK, Mohammed ZR, Hadi S. An efficient method for encryption and hiding any file in color image. Journal of Engineering and Applied Sciences. 2018;13(Special Issue 8):6359-6364.
32. Subramanian N, Elharrouss O, Al-Maadeed S, Bouridane A. Image steganography: A review of the recent advances. IEEE Access. 2021;9:23409-23423. doi:10.1109/ACCESS.2021.3053998
33. USC-SIPI. The USC-SIPI Image Database, Volume 3: Miscellaneous. Signal and Image Processing Institute, University of Southern California; 2024. Available from: https://sipi.usc.edu/database/
34. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP. Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing. 2004;13(4):600-612. doi:10.1109/TIP.2003.819861
35. Westfeld A, Pfitzmann A. Attacks on steganographic systems. Information Hiding (IH 1999), Lecture Notes in

Computer Science, 1768: 61-76; 2000. doi:10.1007/10719724_5
36. Wu DC, Tsai WH. A steganographic method for images by pixel-value differencing. Pattern Recognition Letters. 2003;24(9-10):1613-1626. doi:10.1016/S0167-8655(02)00402-6
37. Wu Y, Noonan JP, Agaian S. NPCR and UACI randomness tests for image encryption. Cyber Journals: Journal of Selected Areas in Telecommunications, April. 2011;2011:31-38.

How to Cite This Article

Mohammed ZR, Kadhim MH. A Finite-Precision robust Crypto-Steganographic scheme for color images using perturbed 2D-LSCM and chaotic pixel permutation. Int J Eng Comput Appl. 2026;2(6):1-18.
doi:10.54660/.IJECA.2026.2.6.01-18.

Creative Commons (CC) License

This is an open access journal, and articles are distributed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0) License, which allows others to remix, tweak, and build upon the work non-commercially, as long as appropriate credit is given and the new creations are licensed under the identical terms.