Journal of Science and Technology – Smart Systems and Devices
Volume xx, Issue x, Month, 2026, ppp-ppp

Incorporating Zero-Knowledge Proofs into Chaos-Based Steganography for Image Verification

Do Tran Quang, Minh Dang Nhat, Khanh Nguyen An, Hue Ta Thi Kim*
Hanoi University of Science and Technology, Ha Noi, Vietnam
Corresponding author email: hue.tathikim@hust.edu.vn

Abstract
Conventional steganographic systems primarily aim at imperceptible data hiding but lack a formal mechanism to verify that an embedding procedure has been executed correctly without revealing the hidden content. This paper proposes a fragile verifiable steganographic framework that integrates chaos-based Least-Significant-Bit (LSB) embedding with a zero-knowledge proof system. A secret message is embedded into the LSBs of image pixels whose locations are permuted using a combination of Arnold's Cat Map and the Logistic Map, with chaos seeds derived from image-dependent features. In parallel, a zk-SNARK (Groth16) proof is generated to attest, in a zero-knowledge proof technique, that a valid LSB embedding consistent with the prescribed chaotic transformations and system parameters has been performed. Verification properties, including chaos parameters, hash commitments, proof length, and the serialized Groth16 proof, reside entirely in pixel LSBs, with the public verification header placed at positions derived from a publicly known proof key. This allows a verifier, given only a single received image and a pre-installed system configuration, to extract the required verification parameters directly from the LSBs of the image and validate the proof without accessing the embedded message. Consequently, any post-embedding modification, including lossy transformations, is treated as an integrity violation that invalidates verification. Experimental results show that the proposed scheme preserves high visual fidelity, achieving favorable PSNR, SSIM, and MSE values while maintaining a Groth16 proof in raw binary of 256 Bytes and a compact proof size of 739 Bytes in the JSON serialization used by the implementation. These properties make the system suitable for deployment in resource-constrained environments, including IoT networks and embedded communication settings such as CAN-based systems, where verifiable embedding integrity is required.

Keywords: Chaos-based cryptography, information hiding, steganography, zero-knowledge proofs, zk-SNARKs.

1. Introduction

Steganography aims to conceal information within digital media while preserving perceptual quality and avoiding detection. In image steganography, spatial-domain least significant bit (LSB) embedding remains attractive for its simplicity and low computational cost, but it is also known to be vulnerable to statistical and visual steganalysis when embedding locations or bit patterns are insufficiently dispersed. To mitigate this, a substantial line of work uses chaotic maps to pseudo-randomize embedding positions and to mask payload bits, leveraging the sensitivity and mixing properties of maps such as Arnold's Cat Map (ACM) and the Logistic map [1–3]. Beyond steganalytic resistance, chaos-based position generation offers a practical advantage over standard block ciphers such as AES for image applications: chaotic maps rely on simple arithmetic operations (multiplication, addition, and modular reduction) that incur low computational overhead for pixel-level processing [4, 5]. This property becomes critical when the embedding computation must be proven inside a zk-SNARK circuit, since AES S-boxes and bitwise XOR operations are expensive to express as Rank-1 constraints, whereas chaotic map iterations translate directly and efficiently into R1CS form. In parallel, zero-knowledge proof (ZKP) systems enable a prover to convince a verifier that a statement is true without revealing any additional information, with zk-SNARKs providing compact proofs and fast verification [6]. The combination of steganography and ZKP has also been explored for ownership-style verification in prior work [7].

Most chaos-based LSB steganography schemes focus on imperceptibility and resistance to steganalysis, but they typically lack a formal cryptographic mechanism for a third-party verifier to confirm that a specific embedding procedure was executed correctly. While prior works have explored combining ZKP with image processing, they often suffer from prohibitive computational costs when proving standard cryptographic primitives (e.g., standard hash functions or AES S-boxes) inside the circuit. The core novelty of this paper lies in demonstrating that chaotic dynamical systems (specifically ACM and the Logistic map) are not merely tools for statistical pseudo-randomness, but they serve as an optimal, algebraic "missing link" for zero-knowledge steganography. Because chaotic maps rely on simple arithmetic operations that translate highly efficiently into Rank-1 Constraint Systems (R1CS), our approach achieves a significantly faster and smaller proof compared to traditional cryptographic counterparts.

Our proposed work presents a "Fragile Verifiable Steganography" framework that tightly couples (i) chaos-controlled LSB embedding driven by ACM and a Logistic-map keystream, and (ii) a Groth16

p-ISSN 3093-3285
e-ISSN 3093-3315
https://doi.org/10.51316/jst.xxx.ssad.2026.36.x.x
Received: Jan 19, 2026; Revised: May 22, 2026;
Accepted: May 29, 2026; Online: Jul 7, 2026

zk-SNARK proof that attests, in zero knowledge, to the correctness of the embedding computation. The term "fragile" is embraced intentionally: the system is specifically designed for high-integrity authentication and tamper detection rather than robustness against lossy transformations. By providing perfect tamper detection, this framework is tailored for lossless, high-integrity communication channels where bit-level fidelity is guaranteed, such as medical LANs or industrial CAN buses. On the steganography side, ACM deterministically generates an ordered set of embedding positions, while a Logistic-map keystream masks payload bits. On the ZKP side, the prover generates a Groth16 proof demonstrating that the embedded payload and positions are consistent with the prescribed chaos-controlled embedding algorithm.

At a high level, the Groth16 instance proves an existential statement of the form:

$$\begin{aligned} &\exists\,(m, K_{\text{steg}}, P, \text{aux}) \text{ s.t.} \\ &\Big(I^* = \mathsf{Embed}(I; m, K_{\text{steg}}, P)\Big) \wedge \\ &\Big(C_{\text{pos}} = \mathsf{MerkleRoot}(H(P))\Big) \wedge \\ &\Big(a = (H(I^*), C_{\text{pos}}, \ldots)\Big), \end{aligned} \tag{1}$$

where I^* is the received stego-image, P is the (collision-free) position sequence generated under ACM parameters, and a denotes public inputs. Concretely, our implementation encodes public data, including an image hash and the Merkle root of positions (and timing/protocol metadata as needed), while keeping embedding positions, keystream material, and serialized proof bits in the witness.

A key design requirement is that verification should succeed given only a single received image plus a pre-installed configuration (e.g., the verification key and parsing rules). In our design, all public parameters required for verification that are chaos parameters, hash commitments, the proof length, and the serialized Groth16 proof are embedded entirely in pixel LSBs at positions derived from a publicly known proof key (a fixed default string, e.g., `zkproof_key_v1`, distinct from the secret chaos key used for the hidden payload). During extraction, the verifier uses this known proof key to run the same Arnold Cat Map walk, regenerates the proof-key-derived positions, and extracts the verification header and embedded proof directly from pixel LSBs. No dedicated PNG chunk or file-level marker is used; to a passive observer, the stego image is an ordinary PNG file with no structural indicators of embedded data. The verifier then reconstructs (π_A, π_B, π_C) and runs the Groth16 pairing check. We further bind the statement to the transmitted image via a public hash $H(I)$ and bind it to the embedding schedule via C_{pos}, so that modifications to the carrier image or the schedule invalidate verification. Groth16 requires a trusted setup for the circuit family. Accordingly, we keep the circuit small and prove only what is necessary for embedding correctness. In our threat model, steganographic keys and the witness remain secret, while the Groth16 verification key is public, and toxic waste is assumed securely discarded after setup. Setup artifacts are distributed out-of-band once, after which verification can be repeated many times at low cost.

The remainder of this paper is organized as follows: Section 2 presents some preliminaries and the description of the process framework for images is provided. Performance evaluation is provided in section 3. Paper ends with some conclusions in section 5.

2. Embedding zk-SNARKs with Chaos-based Steganography

The framework comprises two cooperating subsystems: a chaos-based steganographic system for payload embedding, and a Groth16 zk-SNARK for non-interactive integrity attestation. In this way, Groth16—a zk-SNARKs family—is chosen for its small proof size and fast verification while providing soundness and zero-knowledge guarantees. On the prover side, combining an Arnold Cat Map (ACM) and a Logistic Map drives chaos-based LSB embedding, producing a steganographic image that preserves the cover's appearance. The Groth16 zk-SNARK proves that the embedded payload matches the computation. We choose Groth16 for its three-element proof and single pairing verification. To embed all verification data without file-level markers, both the public metadata (chaos parameters, hash commitments, proof length) and the serialized proof bits are placed in pixel LSBs. The public verification header and the embedded Groth16 proof occupy positions derived from the publicly known proof key via the Arnold Cat Map walk; the secret message payload occupies separate positions derived from the private chaos key.

2.1. Groth16 zk-SNARK Implementation

In the first step, the system encodes the statement to contain public data for the image hash, the Merkle root of positions, and timing metadata. The witness collects private embedding positions, keystream material, and serialized proof bits. The circuit compiled into Rank-1 Constraint System (R1CS)[8] with wire vector $z = (1, \mathbf{a}, \mathbf{w})$ present in

$$(Az) \circ (Bz) = Cz, \tag{2}$$

where $\circ$ denotes the Hadamard product. The R1CS is then processed by the Groth16 proving system to produce the final proof.

Position commitment. A Merkle root equation is computed C_{pos} over canonical encodings of positions $\{p_i\}$:

$$C_{\text{pos}} = \text{MerkleRoot}\big(H(p_1) \,\|\, H(p_2) \,\|\, \cdots \,\|\, H(p_L)\big), \tag{3}$$

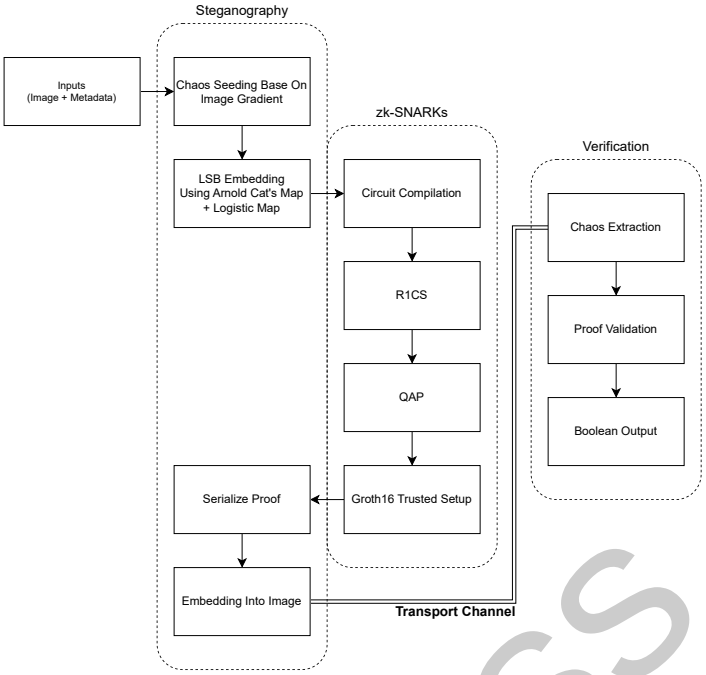


Fig. 1. Embedding ZKP in steganography system model

and expose C_{pos} as a public input.

Proof-bit binding. Proof then serialized into a bitstring $\mathbf{b} \in \{0,1\}^{\ell}$ inside the witness. Verifier expects (π_A, π_B, π_C) to reside in $G_1 \times G_2 \times G_1$ with the correct subgroup checks.

It is crucial to clarify that this process does not verify the content of the secret message (which remains entirely hidden to preserve zero-knowledge properties), nor does it verify the semantic visual content of the image. Instead, it serves to prove the existential statement that: "The current stego-image was produced by correctly applying the chaos-based LSB embedding algorithm to a cover image with SHA-256 hash $H(I)$".

Let $\text{IC}(\mathbf{a}) = vk_0 + \sum_{i=1}^{\ell} a_i\, vk_i \in G_1$. Groth16 verification checks the pairing product

$$e(\pi_A, \pi_B) \stackrel{?}{=} e(vk_\alpha, vk_\beta) \cdot e(\text{IC}(\mathbf{a}), vk_\gamma) \cdot e(\pi_C, vk_\delta), \quad (4)$$

with $vk_\alpha \in G_1$ and $vk_\beta, vk_\gamma, vk_\delta \in G_2$ from the CRS.

The verification header containing chaos parameters, commitments, hashes, and the proof length is embedded in pixel LSBs at positions generated by running the Arnold Cat Map seeded from a publicly known proof key. Since this proof key is fixed and publicly distributed (separate from the secret chaos key), any verifier can regenerate the same positions and extract the verification header without any file-level hints or auxiliary data structures.

2.2. Chaos-based Permutation

This is where Arnold's Cat Map (ACM) is used with (x_0, y_0) inside an $N \times N$ lattice and drives it with a shear matrix parameterized by (a, b) extracted from the chaos key. Each step raises the operator F to the number of rounds rp so that the next embedding coordinate satisfies

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} = F^{rp} \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} (x_i + a \times y_i) \bmod N \\ (b \times x_i + a \times b \times y_i) \bmod N \end{bmatrix}^{rp}. \quad (5)$$

Equation (5) states precisely how the location that hides the next payload bit is computed: starting from (x_i, y_i), apply the keyed shear map F exactly rp times, wrap the result modulo N, and the produced (x_{i+1}, y_{i+1}) is the next pixel. Because the transform is deterministic under (a, b, N, rp), both the prover and the verifier can regenerate the same pseudo-random walk and remain synchronized in embedding order.

Each (x_k, y_k) is mapped to a linear index

$$i_k = (y_k \cdot W + x_k) \bmod (W \cdot H), \quad (6)$$

reject collisions, and obtain an ordered list of unique positions $\{p_1, \ldots, p_L\}$.

A Logistic map supplies the keystream that masks payload bits:

$$x_{n+1} = r x_n (1 - x_n), \qquad x_n \in (0,1), \quad r \in (3.57, 4], \quad (7)$$

a classic chaotic recurrence. When we need additional mixing, coupled form is used:

$$x_{n+1} = \big(f(x_n) + \alpha,\ g(y_n) + \beta,\ h(z_n)\big) \bmod 1, \quad (8)$$

with tunable offsets α, β and companion recurrences for y_n, z_n. Keystream quantization via $k_n = \lfloor 2x_n \rfloor \in \{0,1\}$ and mask payload bits using $b'_n = b_n \oplus k_n$.

LSB replacement is directly applied under chaos control. For a pixel value $v \in \{0,\ldots,255\}$ and a bit $b \in \{0,1\}$,

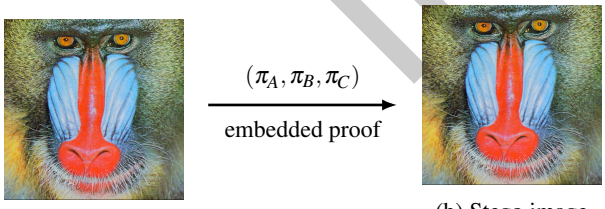
$$v' = v - (v \bmod 2) + b, \tag{9}$$

on the selected channel. The channel index is rotated via $c_n = (x_n + y_n) \bmod 3$ for RGB balance. By keeping the embedding rate low and steering positions with ACM, visible artifacts can be avoided.

2.3. Extraction and Verifiable Embedding

The verifier uses the publicly known proof key to run the Arnold Cat Map walk and derive the proof-key-derived positions. At those positions, the verification header is extracted from pixel LSBs, recovering h_{img} (the SHA-256 hash of the original cover image), C_{pos}, the proof length, and the public ACM parameters (lattice size N, shear coefficients a, b, and round count rp) that define the embedding schedule. Using the recovered chaos parameters, ACM positions and the Logistic keystream are then regenerated, the payload positions $\{p_i\}$ are traversed, and (9) is inverted to recover masked bits. Unmasking follows $b_n = b'_n \oplus k_n$, after which (π_A, π_B, π_C) and the public input vector are reconstructed.

During the Embedding Procedure Integrity Verification, subgroup and structural checks on (π_A, π_B, π_C) are performed. The verifier then executes the pairing test in (4) and returns both the validity flag and the descriptive metadata for audit trails. We emphasize that this checks whether the exact embedding constraints were satisfied on the declared original cover image $H(I)$, proving the integrity of the embedding action itself.



(a) Original cover image

(b) Stego image with embedded proof

Fig. 2. Visual comparison between original cover and generated stego image

Groth16 shows its compact proofs and low verification cost, keeping both payload and CPU requirements modest. We leverage ACM to spread embedding positions across the lattice and pair it with a Logistic keystream to disguise bit patterns under a key. Public metadata is embedded in pixel LSBs at proof-key-derived positions; auditors can extract it by regenerating the proof-key trajectory using the known proof key, without needing to inspect file structure or auxiliary chunks. Finally, bind the circuit to h_{img} and C_{pos} to ensure the proof is image-specific and tamper-resistant even if an adversary gains partial knowledge of the workflow.

3. Experimental Results and Evaluation

In this section, we describe our output benchmark after running the system on various images, including the 'Baboon' test image. There are four criteria for us to consider: imperceptibility and capacity of the LSB embedding; robustness of extraction across different inputs; and end-to-end performance that scales with image size and message length.

Our assessment considers three image categories used commonly in steganography and toolchains: (i) *natural* content (e.g., Baboon 512×512), (ii) *structure-dominant* content (gradient field 768×768), and (iii) *texture-extreme* content (random noise 1024×1024). Two controlled scaling studies are performed: *Image-size scaling* from 128×128 to 1024×1024 at a fixed message; *Message-length scaling* from 50 to 1000 characters at a fixed image size (512×512).

For each image, a uniformly random ASCII message is embedded (or a metadata-derived message) at a fixed payload budget. For the size sweep, the message content is kept constant; for the message-length sweep, the cover is fixed. The correctness of the extraction of the bitwise and the on-disk computation size overhead (PNG) is then confirmed. When applied, distortions are performed post-embedding; extraction then operates on the distorted stego.

3.1. Image Quality Metrics

Image fidelity is measured by MSE, PSNR, and SSIM computed between the cover and stego images per channel and averaged over RGB; these standard metrics are defined in [3]. A lossless pipeline should yield PSNR ≥ 40 dB and SSIM close to 1.0 at low payload rates.

3.2. Pixel Distribution and Entropy Stability

To test imperceptibility at the distribution level, we overlay the per-channel histograms of the cover and stego images. Fig. 3 shows red, green, and blue densities. The curves almost coincide, indicating that bit-plane updates remain sparse and well-scattered. ACM-based positions and the logistic keystream spread flips across textured and flat areas, so global statistics stay steady.

Distribution drift is quantified by per-channel Shannon entropy and Jensen-Shannon divergence (JSD) between cover and stego histograms; both are standard information-theoretic measures [3]. Across runs at low payload ρ, we observe near-zero JSD and negligible entropy change, matching the histogram overlap in Fig. 3 and confirming the high PSNR and SSIM reported earlier. There are two end-to-end variants: *SchnorrZK+Stegano* and *zk-SNARKs+Stegano*. Both

Table 1. Quality & proof metrics

Image/Test Case	PSNR (dB)	SSIM	MSE	Capacity (bpp)	PG (s)	PV (s)	Size (KB)	Overhead (s)
Baboon 512×512	83.38	0.99999998	2.99×10^{-4}	0.00183	0.53	0.51	0.72	+0.53
Random Noise 1024×1024	89.43	0.99999996	7.41×10^{-5}	0.00046	0.54	0.52	0.73	+0.54
Gradient Field 768×768	86.75	0.99999994	1.37×10^{-4}	0.00081	0.54	0.51	0.72	+0.54

Note: Size (KB) refers to the JSON-serialized proof as output by snarkjs. The equivalent raw binary proof (three BN254 group elements) is approximately 256 bytes.

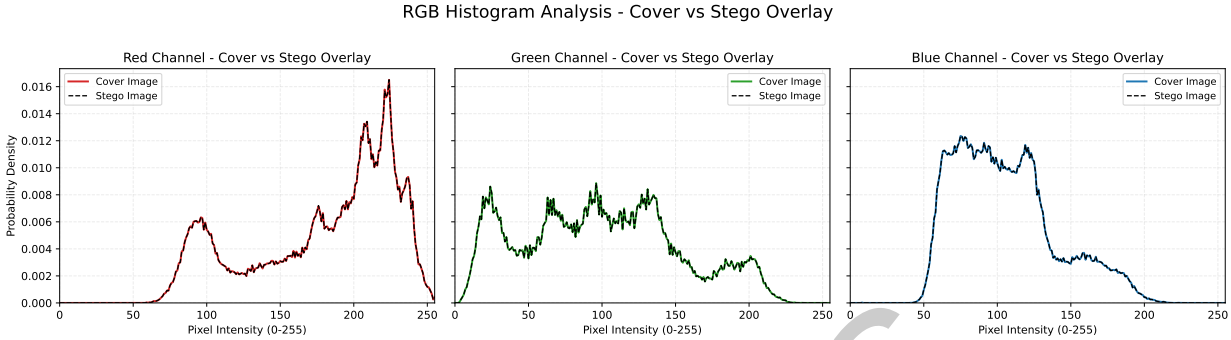


Fig. 3. Histogram comparison between the cover image and stego-image

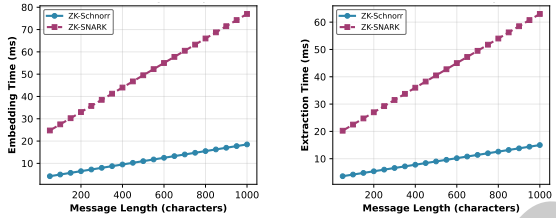


Fig. 4. Performance comparison between zk-SNARKs and SchnorrZK

embed a serialized proof plus public metadata using the same chaos-controlled LSB policy and identical payload rates. Fig. 4 plots embedding and extraction latency versus message length. Schnorr is faster, as expected from its lightweight algebra and simpler verifier.

3.3. Quality Comparison and Steganalysis Resistance

To contextualise the proposed method, we compare it against two simpler baselines at the same payload rate on the Baboon 512×512 image (the primary test case in Table 1): (i) *Sequential LSB*, which embeds bits in raster-scan order with no position randomisation, the standard baseline method in spatial-domain steganography [3]; and (ii) *PRNG-Randomised LSB*, which selects positions using a standard pseudorandom number generator (seed 42), representing the class of non-chaotic randomised embedding methods [1]. All three methods embed the same 180-byte (1440-bit) message at a payload rate of 0.00183 bpp. Two classical steganalysis tests are applied: RS analysis, which reports the ratio $(R_m - S_m)/(R_m + S_m)$ (values near zero indicate no detectable embedding), and a chi-square goodness-of-fit test comparing cover and stego histograms (a p-value near 1.0 indicates indistinguishable distributions). Note that the PSNR values in this table (≈78.5 dB) differ from those in Table 1 (83.38 dB) because the two experiments use different embedding configurations: Table 1 reports the full system pipeline (including Logistic-map keystream masking, which reduces the number of actual pixel modifications when a masked bit matches the existing LSB), whereas this benchmark applies direct LSB replacement without keystream masking to isolate the effect of position selection strategy.

Table 2 shows that all three methods produce nearly identical image quality (PSNR within 0.07 dB, SSIM within 10^{-6}). This is expected: all methods perform single-bit LSB replacement at the same payload rate, so the position selection strategy does not change the number of modified pixels. Classical steganalysis results are likewise indistinguishable across methods. The RS detection ratio is approximately 0.033, equal to the cover image baseline, and the chi-square p-value is 1.0 in every case, confirming that stego histograms are statistically indistinguishable from the cover at this operating rate. Similar results were observed on the gradient-field and random-noise test images, indicating consistency across image types. At the system's operating payload rate of 0.00183 bpp, detectability is governed by the payload rate itself rather than the position selection strategy; all methods are effectively undetectable by classical steganalysis at this rate.

The key differentiator between the methods is therefore not steganalytic resistance but *verifiable embedding integrity*: only the proposed chaos-based method enables generation of a Groth16 zk-SNARK proof attesting to the correctness of the embedding, adding cryptographic verifiability at negligible steganalytic cost. The final row of Table 2 shows that the dominant overhead is Groth16 proof generation (≈0.53 s), not the chaos trajectory computation.

Table 2. Embedding methods comparison: quality & performance

Method	PSNR (dB)	SSIM	RS ratio	χ^2 p-val	JSD	t_{emb} (ms)
Sequential LSB [1]	78.51	0.9999999	0.033	1.00	9.2×10^{-8}	1.1
PRNG-Random LSB [3]	78.57	0.9999977	0.032	1.00	1.2×10^{-7}	2.6
Chaos LSB (Proposed)	78.50	0.9999978	0.033	1.00	9.1×10^{-8}	108.6
Chaos LSB + ZK proof	(same image quality and steganalysis as Chaos LSB)					≈639

3.4. Imperceptibility and Capacity

Performance Scaling: In our image-size sweep (fixed message), embedding time scales *linearly* with pixel count from 4.5 ms at 128^2 to 9.1 ms at 1024^2; memory scales linearly as well (0.02 MB → 13.75 MB). For message-length scaling, runtime increases linearly from 2.7 ms (50 chars) to 40.0 ms (1000 chars). In both studies, the linear model fits with $R^2 \approx 0.99$, confirming the expected $O(n)$ (pixels) and $O(m)$ (message) behavior, and extraction remains constant-time relative to image size (1.8–2.7 ms). The Groth16 pipeline operates end-to-end (witness → proof → verification) with compact public inputs (image hash, commitment root, proof length, timestamp) and a small serialized proof. In our runs, proof generation and verification complete within sub-second latency, and the serialized proof remains sub-kilobyte in size. It means that exact latencies and proof sizes depend on circuit size and serialization format, rather than on environment-specific timings. This overhead is independent of the cover's pixel count when the circuit shape is fixed.

High-Integrity Networks: Rather than evaluating empirical network throughput, we treat specific network conditions as deployment requirements for this fragile steganography system. The framework is intentionally designed for high-integrity, lossless communication channels such as local area networks (LANs) in medical environments or Controller Area Network (CAN) buses in industrial settings. We assume a conservative channel reliability bound of BER $< 10^{-6}$ and rely on transport-layer error correction (e.g., TCP) to ensure lossless delivery. Under these assumptions, the probability of undetected bit-level corruption reaching the application layer is negligible, which is a prerequisite for a system that provides strict verifiable embedding integrity without robustness against lossy edits.

4. Security Analysis

Security goals & Robustness vs. Integrity. We distinguish three primary goals: (G1) *verifiable embedding integrity* (a verifier should accept only if the received image is consistent with the prescribed chaos-controlled LSB embedding and the declared public inputs); (G2) *witness privacy* (the proof should not reveal the secret witness beyond validity); and (G3) *covert payload protection* against unauthorized extraction. Crucially, robustness to lossy transforms is entirely excluded by design. The lack of robustness against lossy compression is treated as a *strict security requirement* for Perfect Tamper Detection. Because we use SHA-256 binding for the original image, any minor alteration will cause the verification to fail, thus providing Strict Integrity.

Threat model and assumptions. We consider (i) a passive adversary who observes the stego image I^* (no auxiliary metadata is attached; all verification data resides in pixel LSBs), and (ii) an active adversary who can modify pixels or tamper with public inputs such as $H(I)$ and C_{pos} embedded within the image LSBs. The stego key material K_{steg} and witness w remain secret, while the Groth16 verification key VK is public. Groth16 requires a trusted setup; we assume toxic waste is securely discarded after setup. If the setup is compromised, an adversary may generate accepting proofs for false statements.

What is proven (high-level statement). Verification checks a Groth16 proof for an existential statement of the form: $\exists w = (m, K_{\text{steg}}, \theta, \ldots)$ such that the observed carrier I^* is consistent with executing the embedding algorithm under chaos parameters θ (Arnold Cat Map schedule and Logistic-map masking) and such that the public inputs extracted from the image match $(H(I^*), C_{\text{pos}}, \ldots)$.

In our implementation, the verifier uses the publicly known proof key to regenerate the proof-key-derived positions, extracts the verification header (including h_{img} and C_{pos}) directly from pixel LSBs at those positions, and then deterministically regenerates the embedding schedule and keystream to recover the serialized proof.

Workflow and Circularity Avoidance. To eliminate any circular verification vulnerabilities, the system adheres to a rigorous sequential workflow: (1) The SHA-256 hash $H(I)$ is computed exclusively from the *Original Cover Image*. (2) The secret message is embedded into the LSBs. (3) The zk-SNARK proof (comprising approximately 15,000 R1CS constraints for the secure version) is generated based on the original Cover $H(I)$ and the selected embedding parameters. (4) Finally, the generated proof is embedded into the image's LSBs. This guarantees that the statement is firmly anchored to the original, unperturbed cover image rather than a cyclic hash of the final stego-image.

Soundness and tamper detection. By Groth16 soundness (under standard assumptions such as discrete-log hardness), no PPT adversary can forge

an accepting proof for a false statement except with negligible probability. Consequently, any modification to the proof bits, the carrier-dependent digest, or schedule commitments (e.g., $H(I)$ and C_{pos}) causes the pairing-based verification to fail. This provides strong integrity: an active attacker can cause rejection (denial-of-service) by distorting pixels, but cannot make a modified image *pass* without producing a fresh valid proof.

No Structural Markers. The framework is explicitly designed to operate without any file-level metadata structures. No PNG chunks (such as custom `zkPF` chunks) are used to store the proof or parameters. Instead, everything resides strictly within the spatial LSBs. This guarantees that no structural markers exist, rendering the stego-image entirely indistinguishable from a standard, untampered PNG file to a passive observer.

5. Conclusion

This paper introduces a verifiable image steganography framework that combines chaos-controlled LSB embedding (via Arnold's Cat Map and the Logistic map) with a Groth16 zk-SNARK attestation. Verification parameters, including hash commitments, proof length, and the serialized Groth16 proof, reside entirely in pixel LSBs at positions derived from a publicly known proof key, so the stego image carries no file-level markers and is indistinguishable from an ordinary PNG by structural inspection. The proof is bound to the received stego-image and the embedding schedule via public digests/commitments (e.g., $H(I)$ and the position commitment), enabling a verifier to validate the embedding procedure without learning the hidden payload. Experimental results indicate that the proposed scheme preserves high visual fidelity at low payload rates, with PSNR ranging from 83.38 to 89.43 dB and SSIM close to 1.0 across representative test images. End-to-end measurements further show that the steganographic embedding cost scales linearly with image size and message length (with $R^2 \approx 0.99$ across the reported sweeps), while extraction remains nearly constant time with respect to image size in our implementation. On the proof side, the system employs a secure circuit with approximately 15,000 R1CS constraints, yet still produces succinct proofs of 739 bytes in the JSON serialization used by the implementation pipeline (256 bytes in raw binary), enabling practical verification under resource constraints. Baseline comparisons against sequential and PRNG-randomized LSB embedding confirm that all three methods are equally undetectable by classical steganalysis (RS analysis and chi-square tests) at the system's operating payload rate. Unlike traditional steganographic or semantic image authentication schemes, the proposed framework is intentionally designed as a fragile verifiable steganographic system in which any post-embedding modification, including lossy transformations, is treated as an integrity violation that invalidates proof verification. Rather than authenticating semantic image content, the framework focuses on verifying the integrity and correctness of the embedding procedure itself. This enables highly sensitive tamper detection and makes the framework particularly suitable for high-integrity, lossless communication channels.

References

[1] M. Mishra, A. R. Routray, and S. Kumar, High security image steganography with modified Arnold's Cat Map, International Journal of Computer Applications, vol. 37, no. 9, Jan. 2012. https://doi.org/10.5120/4636-6685

[2] T. T. K. Hue, N.T.Linh, M. Nguyen-Duc, and T. M. Hoang, Data hiding in bit-plane medical image using chaos-based steganography, pp. 1–6, Oct. 2021. https://doi.org/10.1109/MAPR53640.2021.9585243

[3] N. Subramanian, O. Elharrouss, S. Al-Maadeed, and A. Bouridane, Image steganography: A review of the recent advances, IEEE Access, vol. 9, Jan. 2021. https://doi.org/10.1109/ACCESS.2021.3053998

[4] U. Zia, M. McCartney, B. Scotney, J. Martinez, M. AbuTair, J. Memon, and A. Sajjad, Survey on image encryption techniques using chaotic maps in spatial, transform and spatiotemporal domains, Int. J. Inf. Secur. vol. 21, pp. 917–935, Apr. 2022. https://doi.org/10.1007/s10207-022-00588-5

[5] W. Alexan, N. H. E. Shabasy, N. Ehab, and E. A. Maher, A secure and efficient image encryption scheme based on chaotic systems and nonlinear transformations, Scientific Reports, vol. 15, Aug. 2025. https://doi.org/10.1038/s41598-025-15794-z

[6] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof systems, SIAM J. Comput. vol. 18, no. 1, pp. 186–208, Feb. 1989. https://doi.org/10.1137/0218012.

[7] M. D. Toso and S. Chung, Combining Steganography and Zero-Knowledge Proofs to Embed and Prove a Digital Signature in an Image, in [Online] Available: https://api.semanticscholar.org/CorpusID:220068911, 2006.

[8] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, PLONK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge, in [Online] Available: IACR Cryptology ePrint Archive, Report 2019/953, 2019.