cryptography

MDPI

Article

Hybrid Architecture for Protected Data Communication Inside the Private Cloud

Biswaranjan Senapati [1,*], Lalit Narayan Mishra [2], Awad Bin Naeem [3] and Amit J. Rangari [4]

[1] Department of Computer Science, University of Arkansas at Little Rock (UALR), Little Rock, AR 72204, USA
[2] Lowe's Companies Inc., Mooresville, NC 28117, USA; lalitmishra@ieee.org
[3] Department of Biomedical and Neuromotor Sciences, University of Bologna, Via Zamboni, 33, 40126 Bologna, Italy; awadbinnaeem@gmail.com
[4] JPMorgan Chase, Atlanta, GA 30326, USA; amitrangari@ieee.org
* Correspondence: bsenapati@ualr.edu

Abstract

Private cloud object stores provide infrastructure isolation but leave application-layer data exposed to insider threats and compromised credentials. This paper presents an engineering integration of an Add-Rotate-XOR (ARX) block cipher and multi-bit Least Significant Bit (LSB) steganography into an end-to-end pipeline for private MinIO object storage. The cipher, KREA v2, is a SPECK-64/128 derived ARX construction with three application-driven choices: CRC32 key whitening, byte-aligned rotations ($\alpha = 7$, $\beta = 2$), and deterministic CTR-mode nonces. Mixed Integer Linear Programming (MILP) trail analysis matches SPECK-64/128's minimum-trail weights through rounds 1–4. KREA v2 ciphertext meets standard keystream-quality preconditions (NIST SP 800-22 battery, 49.98% mean avalanche, Shannon entropy 7.9992–7.9998 bits/byte across realistic XML, JSON, video, and HTTP/2 payloads). Modified LSB (MLSB) embeds 3 bits per RGB channel with an XOR watermark at 37–38 dB Peak Signal-to-Noise Ratio (PSNR), providing 3× standard-LSB capacity. Steganalysis uses chi-square and RS detectors plus a Convolutional Neural Network (CNN) detector (Yedroudj-Net) trained on 8000 BOSSBase-1.01 cover/stego pairs; CNN area under the ROC curve is $\geq$0.999 against the watermarked variant. The MinIO pipeline runs at 355.1 ms (68.6% network I/O) with 100% message fidelity. The XOR watermark increases RS detectability above 75% capacity; a 200-image ablation cuts median RS detection (0.289 to 0.000) and mean (0.342 to 0.130) in a sparse-keystream variant, prioritised for follow-on full-scale evaluation. The architecture is offered as a documented engineering integration with explicit security caveats and threat-model boundaries, not as a production-hardened cryptographic primitive.

Keywords: ARX block cipher; steganography; private cloud security; modified LSB; steganalysis resistance; MinIO object storage; hybrid cryptography-steganography

Academic Editor: Jim Plusquellic

Received: 15 April 2026
Revised: 25 May 2026
Accepted: 30 May 2026
Published: 2 June 2026

1. Introduction

1.1. Background and Motivation

Private cloud deployments have become the infrastructure of choice for organizations handling sensitive workloads: financial records, healthcare data, proprietary source code, and regulated communications. Unlike public clouds, where compute resources are shared across tenants, a private cloud dedicates its infrastructure to a single organization, typically enforced by network segmentation and firewall policy [1]. Yet infrastructure isolation does

not, by itself, guarantee data confidentiality. Data at rest within the private cloud remains exposed to insider threats, compromised service accounts, and lateral-movement attacks after an initial perimeter breach [2,3]. Cryptographic controls at the transport layer (TLS) protect point-to-point connections, but once encrypted data lands in an object store, it is stored in plaintext unless application-layer encryption is explicitly applied [4].

The standard response to this gap is symmetric encryption at the application layer, and well-analyzed ciphers such as AES-128 [5] or ChaCha20 handle this role effectively in high-throughput environments. Lightweight ciphers designed for constrained devices, including PRESENT [6], SIMON and SPECK [7], and Ascon (the NIST Lightweight Cryptography winner) [8], address environments where AES hardware acceleration is unavailable. Private cloud object stores occupy an intermediate position: they are not resource-constrained like embedded devices, but they serve high-concurrency streaming workloads where per-object encryption overhead accumulates at scale. A cipher designed specifically for the steganographic application context can incorporate structural properties of the carrier channel, such as pixel-aligned block boundaries and context-bound key whitening, that general-purpose ciphers do not include. We position the resulting construction as a parameterized adaptation of SPECK-64/128 [7] that inherits SPECK's analyzed security properties rather than as a new cryptographic primitive subject to fresh adversarial scrutiny; the contribution of this paper is the integrated pipeline and its measured behavior in a private cloud deployment, not a security advance over SPECK or AES.

Steganography addresses a complementary security objective. While encryption transforms the content of a message into an unintelligible form, steganography conceals the existence of the message itself by embedding it within an innocuous carrier, typically a digital image [9]. Combining the two techniques implements defense-in-depth: an adversary intercepting a stego image must first detect that a hidden payload exists (defeating steganographic security), then decrypt the recovered bytes (defeating cryptographic security). This layered approach has been studied for general network channels [10–12], but its application inside private cloud object stores, where both the upload mechanism and the retrieval API are under organizational control, remains underexplored. When the storage platform is MinIO or a comparable S3-compatible system, the API surface provides deterministic latency and lossless byte preservation, both of which are prerequisites for reliable steganographic retrieval that are not guaranteed over public channels.

1.2. Problem Statement

Existing hybrid crypto-steganography architectures for cloud environments lack three properties simultaneously. First, a cipher whose design accounts for the statistical properties of steganographic payloads rather than treating them as generic byte strings. Second, a steganographic embedding scheme that offers quantifiable capacity advantage over standard Least Significant Bit (LSB) replacement while maintaining measurable steganalysis resistance. Third, an end-to-end system demonstration on a self-hosted private cloud with measured latency and fidelity. Papers addressing cloud data protection typically evaluate either encryption or steganography in isolation, or combine them without characterizing the interaction between cipher output entropy and embedding detectability. This work addresses that gap by co-designing the cipher and the embedding scheme so that their properties reinforce each other within a private cloud deployment.

1.3. Contributions

This paper makes three concrete contributions:

1. KREA v2, a parameterized ARX (Add-Rotate-XOR) block cipher adapted from SPECK-64/128 [7] with three steganography-specific modifications: context-binding key

whitening, modified rotation constants ($\alpha = 7$, $\beta = 2$), and deterministic CTR-mode nonce derivation. KREA v2 passed all 15 NIST SP 800-22 [13] tests, achieved 49.98% mean avalanche effect (320,000 trials), and produced ciphertext with Shannon entropy of at least 7.9992 bits/byte across both baseline and realistic payload classes.

2. MLSB (Modified Least Significant Bit), a steganographic embedding scheme that distributes secret data across the 3 LSBs of each RGB channel per pixel and applies an XOR watermark layer before embedding. MLSB provides 3× the capacity of standard single-bit LSB replacement. The XOR pre-processing step disrupts the pair-of-values histogram patterns that chi-square steganalysis exploits, and the scheme is evaluated against standard LSB and Pixel Value Differencing (PVD) [14] at five embedding densities, with corrected PSNR measurements that account for embedding rate when comparing methods.
3. An end-to-end hybrid architecture that integrates KREA v2 encryption, MLSB embedding, and MinIO private cloud object storage into a single pipeline. The architecture is evaluated on round-trip latency (under 700 ms for messages up to 8 KB) and message fidelity (100% byte-exact recovery across all tested configurations), with benchmark comparisons against AES-128 and Blowfish [15] for the encryption component.

1.4. Paper Organization

Section 2 reviews related work on lightweight block ciphers, image steganography, and hybrid crypto-steganographic systems, with explicit identification of the gaps this work addresses. Section 3 defines the threat model and security goals. Section 4 describes the system architecture, covering the formal specification of KREA v2, the MLSB algorithm, and the MinIO deployment. Section 5 presents the security analysis of KREA v2 and MLSB steganalysis resistance. Section 6 reports experimental results: steganographic quality, capacity, steganalysis detection, and end-to-end pipeline performance. Section 7 discusses findings and limitations, and Section 8 concludes the paper.

2. Related Work

Three bodies of prior work bear directly on the contributions of this paper: lightweight block cipher design, image steganography with multi-bit embedding, and hybrid cryptography-steganography systems deployed in cloud environments. We review each in turn, identifying the specific gap that motivates the present work.

2.1. Lightweight Block Ciphers

The field of lightweight cryptography originated from the need to protect constrained environments, such as embedded sensors and IoT devices, where AES-128 imposes unacceptable memory or latency overhead. PRESENT [6], introduced by Bogdanov et al. at CHES 2007 and later standardized as ISO/IEC 29192-2 [16], established the foundational design paradigm: a substitution-permutation network (SPN) operating on 64-bit blocks with an 80-bit key over 31 rounds, using a 4-bit S-box chosen for low maximum differential probability and linear bias. PRESENT demonstrated that strong cryptographic properties are achievable at gate counts well below AES.

SIMON and SPECK [7], published by Beaulieu et al. at the NSA in 2013, took a different approach. SIMON is an AND-RX design suited to hardware, while SPECK is an ARX (Add-Rotate-XOR) design optimized for software and microcontroller implementations. SPECK-64/128 performs 27 rounds on 64-bit blocks with a 128-bit key; its three-operation inner loop (modular addition, bitwise rotation, XOR) achieves high throughput on platforms with native 32-bit or 64-bit word operations. Subsequent families continued this trajectory. GIFT [17] (Banik et al., CHES 2017) improved upon PRESENT's bit permutation

layer to reduce implementation cost while maintaining the 128-bit security level. The LEA cipher [18] standardized ARX operations with a 128-bit block size as a Korean national encryption standard (KS X 3246 [19]). Ascon [8], selected as the NIST Lightweight Cryptography winner in 2023, uses a duplex-sponge construction to provide authenticated encryption with associated data (AEAD), consolidating confidentiality and integrity in a single primitive.

A common thread across PRESENT, GIFT, SIMON, and SPECK is that they were designed for general constrained environments and carry no application-specific structure. Our cipher, KREA v2, adapts the SPECK ARX design with three steganography-specific modifications (context-binding whitening, modified rotation constants, and deterministic CTR nonce derivation), detailed in Section 4.

2.2. Image Steganography

Standard LSB steganography [20], which replaces the single least significant bit of each pixel channel with one message bit, has been studied since the late 1990s. Its principal limitation is well-documented: chi-square analysis [21] and RS analysis [22] can detect the statistical regularity introduced by 1-bit replacement at typical embedding rates, making standard LSB unsuitable for adversarial settings. Multi-bit LSB variants (2 to 4 bits per channel) increase capacity but compound the detectability problem, since larger modifications produce more pronounced histographic and pixel-pair artifacts.

Pixel-Value Differencing (PVD), introduced by Wu and Tsai [14], takes an adaptive approach: it embeds more bits in smooth regions and fewer bits in textured regions by measuring the absolute difference between adjacent pixel pairs. PVD achieves reasonable imperceptibility at moderate capacity but remains vulnerable to the PVD histogram attack, which exploits the characteristic stepped distribution of inter-pixel differences in stego images. Content-adaptive methods developed in the following decade raised detection resistance substantially. HUGO (Pevny et al., 2010) [23] and WOW (Holub and Fridrich, 2012) [24] embed in directions that maximally disturb the cover image's higher-order statistics, measured against a rich model of the image. S-UNIWARD (Holub et al., 2014) [25] extends this by computing a distortion function in the undecimated wavelet domain, consistently outperforming both HUGO and WOW against the Spatial Rich Model (SRM) feature set [26] and modern CNN-based detectors. S-UNIWARD represents the current state of the art for spatial-domain detection resistance.

MLSB occupies a deliberately different position in this design space. It does not attempt to match S-UNIWARD's detection resistance, which requires complex distortion computation and per-image cost-function evaluation. Instead, MLSB prioritizes practical capacity and implementation simplicity: embedding 3 bits per channel delivers 3× the throughput of standard LSB and integrates directly with KREA v2's 64-bit block output without padding overhead. The XOR watermark layer, applied over the embedded bits using a shared key, disrupts the fixed statistical pattern that chi-square and RS attacks require, providing partial steganalysis resistance measurable against these classical detectors. MLSB is thus positioned as a capacity-optimized steganographic layer with XOR watermark-based partial steganalysis resistance, suited to controlled private cloud deployments where the adversary model does not include access to modern content-adaptive detectors.

2.3. Hybrid Crypto-Steganography for Cloud Security

Several recent works combine encryption with steganography for cloud data protection. Adee and Mouratidis [1] proposed a four-step data security model for cloud data, integrating cryptography and steganography as consecutive layers, and demonstrated end-to-end security on a simulated cloud environment. Their work confirms the architectural

value of the dual-layer approach but relies entirely on standard AES for the cipher layer and conventional LSB for embedding; no custom cipher design or steganalysis evaluation is provided.

Bokhari and Martínez Herráiz [27] proposed a hybrid system combining RSA key exchange, AES encryption, and standard LSB steganography for hybrid cloud communication. While their system targets the same threat model (interception during cloud transit), it uses established NIST-standardized ciphers and 1-bit LSB embedding, omitting both cipher customization and steganalysis robustness testing. Yadav and Singh [28] combined bioinspired elliptic curve cryptography (BECC) using the Mayfly optimization algorithm with a modified LSB scheme, demonstrating that non-standard key generation strategies can improve resistance to brute-force attacks. However, their evaluation is limited to image quality metrics (PSNR, Structural Similarity Index Measure (SSIM)) with no steganalysis or cloud deployment evaluation.

The pattern is consistent: prior hybrid systems either use standard ciphers (AES, RSA) with no cipher-level contribution, or propose steganographic improvements without a custom cipher. None report end-to-end evaluation through a real private cloud deployment with measured round-trip latency. Our system addresses all three gaps simultaneously. Recent contributions in this design space include LSB-class steganography combined with hybrid AES/Blowfish encryption motivated by cloud storage scenarios [29], chaotic-map encryption layered with image-to-audio format conversion for cloud image transfer [30], and learned generative steganography for edge-cloud computing environments [31]; these three directions explore orthogonal axes (LSB-plus-standard-cipher, format-conversion-as-stego, and generative-model-based stego, respectively) and remain distinct from the cipher-codesigned approach pursued in the present work.

2.4. Summary and Research Gap

The literature establishes strong foundations in each component area: ARX ciphers with good lightweight properties (SPECK, GIFT), high-capacity spatial steganography (multi-bit LSB, PVD), and hybrid crypto-steganography architectures for cloud [27,28]. What no published work provides is an integrated system that combines (1) a custom ARX cipher with application-specific structural choices, (2) multi-bit steganography with an XOR watermark anti-detection layer, and (3) measured end-to-end evaluation through a real private cloud deployment. Table 1 summarizes this gap across the most directly comparable works.

Table 1. Comparison of related hybrid crypto-steganography systems.

Paper	Custom Cipher?	Multi-Bit Stego?	XOR Watermark?	Cloud Eval?	E2E Latency?
Adee & Mouratidis (2022) [1]	No (AES)	No (1-bit)	No	Simulated	No
Bokhari & Martínez Herráiz (2024) [27]	No (AES+RSA)	No (1-bit)	No	Simulated	No
Yadav & Singh (2024) [28]	No (BECC)	Yes	No	Not specified	No
This work	Yes (ARX)	Yes (3-bit)	Yes	Real (MinIO)	Yes

This paper fills the identified gap with three contributions: KREA v2, a SPECK-derived ARX cipher with context-binding whitening and pixel-aligned block structure; MLSB, 3-bit embedding with an XOR watermark for chi-square and RS steganalysis resistance; and an end-to-end evaluation pipeline on a MinIO-based private cloud with full latency breakdown and 100% message fidelity verification.

3. Threat Model and Security Goals

3.1. Adversary Model

We define two adversary classes that the hybrid pipeline must withstand.

A **passive network adversary** can observe all data transmitted between the sender node, the MinIO private cloud, and the receiver node. This adversary captures TLS-decrypted objects stored in the cloud (for example, by compromising an object-storage credential or a cloud administrator account), inspects the byte contents of stego images, and performs offline statistical analysis on those images. The passive adversary does not alter transmitted data; the threat is disclosure rather than tampering.

An **active steganalysis adversary** mounts automated steganalysis against stego images after retrieval. This adversary applies standard statistical attacks, specifically chi-square analysis [21] and RS (Regular-Singular) analysis [22], in an attempt to detect the presence of hidden data. A successful steganalysis attack does not immediately recover the hidden message, but it reveals that covert communication is occurring, which may constitute a security failure in contexts where undetectability is required.

The adversary model explicitly excludes the following threat classes. First, it excludes adversaries with physical access to sender or receiver machines; key material and plaintext exist only on these endpoints, and physical compromise is outside the scope of a software architecture paper. Second, it excludes side-channel attacks against the cipher implementation, such as timing analysis or cache-timing attacks on the ARX round function; mitigating these requires hardware-level countermeasures beyond the scope of this work. Third, it excludes adversaries who can modify images in transit (a man-in-the-middle with write access to the TLS channel), as TLS 1.3 provides integrity guarantees for data in transit. Compute-side adversaries with co-resident or microarchitecturally adjacent access are increasingly addressed by Trusted Execution Environment (TEE) and confidential computing primitives [32], with recent systematizations focusing in particular on TEE-protected machine-learning workloads [33] and edge-to-cloud confidential computing for ML pipelines [34]; this paper's contribution is complementary, addressing the storage-tier confidentiality and undetectability gap that persists even when the compute path is TEE-protected.

3.2. Security Properties

The system targets three formal security properties.

Confidentiality. An adversary who obtains the stego image, including the full ciphertext embedded within it, cannot recover the original plaintext message without possessing the 128-bit symmetric key K. Formally, we require that KREA v2 in Counter (CTR) mode achieves indistinguishability under chosen-plaintext attack (IND-CPA). The CTR mode construction reduces this to the pseudorandomness of the underlying KREA block cipher, following the standard CTR mode security argument [35]. This reduction assumes that no nonce value is reused across distinct messages encrypted under the same key; the deterministic nonce derivation scheme (Section 4) is designed to ensure this property.

Undetectability (design aspiration, scope-limited). Ideally, a stego image produced by the MLSB embedding procedure would be statistically indistinguishable from an unmodified natural photograph under automated steganalysis. The XOR watermark layer was designed to disrupt the structured pixel-value distributions that chi-square and RS steganalysis detect, and Section 5.9 reports the empirical outcome against those classical detectors. Against modern CNN-based steganalyzers (Section 5.8), the present XOR-watermark variant does not achieve indistinguishability at high embedding densities; the corresponding operating-regime restriction and threat-model boundary are stated in Section 7.3.

https://doi.org/10.3390/cryptography10030036

Fidelity. The receiver must recover a bit-exact copy of the original plaintext message. No corruption, truncation, or bit-flip errors are acceptable. The length-prefixed message format employed by KREA v2 CTR mode ensures that the receiver decrypts exactly the number of bytes present in the original message; 100% fidelity is a hard requirement and is verified end-to-end in the evaluation. Note that this is a data fidelity requirement, not a cryptographic integrity guarantee: KREA v2 does not include an authentication tag (Message Authentication Code (MAC) or AEAD), so tampered ciphertext would decrypt to corrupted plaintext without detection. Adding authenticated encryption (e.g., Galois/Counter Mode (GCM) or a Hash-based Message Authentication Code (HMAC) layer) is a straightforward extension left for future work.

3.3. System Assumptions

The security analysis rests on three assumptions that must hold for the stated properties to be achieved.

First, the symmetric key K (128 bits) is pre-shared between sender and receiver via a secure out-of-band channel. Key establishment is outside the scope of this work. In practice, Diffie-Hellman key exchange or a public-key infrastructure can fulfill this requirement.

Second, cover images used for steganographic embedding are natural photographs, meaning they are not synthetic images generated by a computer program. Natural photographs contain the smooth tonal gradations and spatial correlations that make LSB-family embedding difficult to detect; synthetic images (solid fills, gradients, programmatically generated graphics) have distinct statistical properties that may interact unpredictably with the embedding procedure.

Third, the MinIO private cloud instance enforces TLS 1.3 for all HTTPS connections used to upload and download objects. Data at rest in MinIO object storage is assumed to reside on infrastructure controlled by the organization, not a shared public cloud provider, consistent with the private cloud deployment model described in the introduction.

4. Proposed Architecture

4.1. System Overview

The architecture implements a three-tier pipeline, illustrated in Figure 1: (Tier 1) sender-side encryption and embedding, (Tier 2) private cloud object storage, and (Tier 3) receiver-side extraction and decryption.

On the sender node, the plaintext message M is first encrypted using KREA v2 in CTR mode (Algorithm 1, Algorithm 2, Algorithm 3) to produce a ciphertext bitstream B. The bitstream B is then embedded into a natural cover image I using the MLSB procedure (Algorithm 4), producing a stego image S. The sender uploads S to the MinIO private cloud over HTTPS with TLS 1.3. The stego image is stored as a PNG file, which uses lossless compression and therefore preserves the embedded bit pattern exactly through the storage and retrieval cycle.

On the receiver node, the stego image S is downloaded from MinIO over HTTPS. The MLSB extraction procedure (Algorithm 5) recovers the ciphertext bitstream B from S. KREA v2 decryption then recovers the original plaintext message M from B. The round-trip process requires no out-of-band communication other than the pre-shared key K and knowledge of the watermark key W (which can be derived from K as described in Section 4.5).

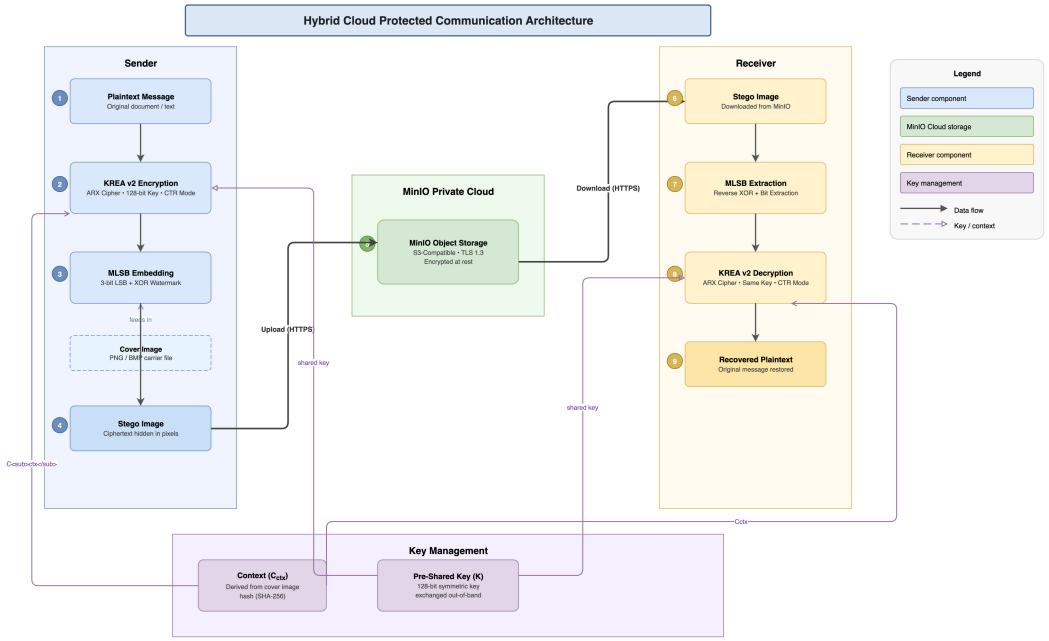

Figure 1. Three-tier hybrid cloud communication architecture. Tier 1 (Sender): KREA v2 encryption followed by MLSB embedding into a cover image. Tier 2 (Cloud): MinIO object storage with TLS 1.3 transport. Tier 3 (Receiver): MLSB extraction followed by KREA v2 decryption. Key management provides the pre-shared key and context binding.

Algorithm 1 KREA-Encrypt

Require: Plaintext block (x, y) where x, y are 32-bit words; round keys $k[0], k[1], \ldots, k[26]$; context value C_{ctx} (32-bit)
Ensure: Ciphertext block (x, y)
Pre-whitening (context binding):
$x \leftarrow x \oplus \text{ROR}(C_{\text{ctx}}, 11)$
$y \leftarrow y \oplus \text{ROL}(C_{\text{ctx}}, 5)$
Round function:
for $i = 0$ to 26 **do**
    $x \leftarrow (((x \ggg 7) + y) \bmod 2^{32}) \oplus k[i]$
    $y \leftarrow (y \lll 2) \oplus x$
end for
Post-whitening (context binding):
$x \leftarrow x \oplus \text{ROL}(C_{\text{ctx}}, 7)$
$y \leftarrow y \oplus \text{ROR}(C_{\text{ctx}}, 13)$
return (x, y)

where $\ggg$ denotes right rotation, $\lll$ denotes left rotation, $+$ denotes addition modulo 2^{32}, and $\oplus$ denotes bitwise exclusive-or.

Algorithm 2 KREA-Decrypt

Require: Ciphertext block (x, y) where x, y are 32-bit words; round keys $k[0], k[1], \ldots, k[26]$; context value C_{ctx} (32-bit)
Ensure: Plaintext block (x, y)
Reverse post-whitening:
$x \leftarrow x \oplus \text{ROL}(C_{\text{ctx}}, 7)$
$y \leftarrow y \oplus \text{ROR}(C_{\text{ctx}}, 13)$
Reverse round function:
for $i = 26$ down to 0 **do**
    $y \leftarrow (y \oplus x) \ggg 2$
    $x \leftarrow (((x \oplus k[i]) - y) \bmod 2^{32}) \lll 7$
end for
Reverse pre-whitening:
$x \leftarrow x \oplus \text{ROR}(C_{\text{ctx}}, 11)$
$y \leftarrow y \oplus \text{ROL}(C_{\text{ctx}}, 5)$
return (x, y)

Algorithm 3 KREA-KeySchedule

Require: Master key K split into $(l[2], l[1], l[0], k[0])$; number of rounds $T = 27$; $m = 3$ (key words minus 1)
Ensure: Round keys $k[0], k[1], \ldots, k[26]$
for $i = 0$ to 25 **do**
  $l[i+m] \leftarrow ((l[i] \ggg 7) + k[i]) \bmod 2^{32} \oplus i$
  $k[i+1] \leftarrow (k[i] \lll 2) \oplus l[i+m]$
end for
return $k[0 \ldots 26]$

Algorithm 4 MLSB Embedding

Require: Cover image C of size $M \times N$ pixels (3 channels: R, G, B); encrypted message bitstream B of length L bits; watermark key (byte array) W of length K_W bytes
Ensure: Stego image S
capacity $\leftarrow M \times N \times 3 \times 3$ ▷ 3 bits per channel, 3 channels
if $L >$ capacity **then**
  abort with capacity error
end if
pos $\leftarrow 0$; idx $\leftarrow 0$ ▷ idx indexes channel values
for each pixel (i, j) where $0 \le i < M, 0 \le j < N$ **do**
  for each channel $c \in \{R, G, B\}$ **do**
    if pos $< L$ **then**
      $v \leftarrow C(i, j, c)$
      $v \leftarrow v$ & 0xF8 ▷ Clear bits 2, 1, 0
      $v \leftarrow v \mid B[\text{pos} : \text{pos} + 3]$ ▷ Embed 3 message bits
      $w_3 \leftarrow W[\text{idx} \bmod K_W]$ & 0x07 ▷ 3 low bits of watermark byte
      $S(i, j, c) \leftarrow v \oplus w_3$ ▷ Apply 3-bit watermark
      pos $\leftarrow$ pos $+ 3$; idx $\leftarrow$ idx $+ 1$
    else
      $S(i, j, c) \leftarrow C(i, j, c)$
    end if
  end for
end for
return S

Algorithm 5 MLSB Extraction

Require: Stego image S of size $M \times N$ pixels; watermark key (byte array) W of length K_W bytes; message length L in bits
Ensure: Recovered encrypted bitstream B'
$B' \leftarrow \emptyset$; pos $\leftarrow 0$; idx $\leftarrow 0$
for each pixel (i, j) where $0 \le i < M, 0 \le j < N$ **do**
  for each channel $c \in \{R, G, B\}$ **do**
    if pos $< L$ **then**
      $w_3 \leftarrow W[\text{idx} \bmod K_W]$ & 0x07 ▷ 3 low bits of watermark byte
      $v \leftarrow S(i, j, c) \oplus w_3$ ▷ Reverse 3-bit watermark
      $b \leftarrow v$ & 0x07 ▷ Extract bits 2, 1, 0
      Append b (3 bits) to B'
      pos $\leftarrow$ pos $+ 3$; idx $\leftarrow$ idx $+ 1$
    end if
  end for
end for
return $B'[0 : L]$ ▷ Trim to exact message length

The architecture achieves layered security: an adversary who intercepts the stego image obtains data protected by two independent mechanisms. The MLSB watermark

layer obscures the presence of embedded data, and the KREA v2 encryption ensures that even if the embedded data is extracted, the ciphertext cannot be deciphered without K.

4.2. KREA v2: ARX Block Cipher

4.2.1. Design Rationale

KREA v2 (Key Round Encryption Algorithm, version 2) adapts the SPECK ARX (Add-Rotate-XOR) design introduced by Beaulieu et al. [7]. ARX designs build the round function entirely from three operations present in all general-purpose processors: modular addition, bitwise rotation, and bitwise exclusive-or. This design choice eliminates the need for S-boxes, making the cipher implementable in high-level languages such as Python without hardware acceleration, and making the diffusion properties analytically tractable through differential and linear cryptanalysis frameworks established for the ARX family.

Three steganography-specific modifications distinguish KREA v2 from the standard SPECK-64/128 cipher from which it derives its structure. First, a context-binding whitening step ties each ciphertext block to the cover image's dimensions, making the same plaintext under the same key produce different ciphertext for different covers; we frame this as a defense-in-depth context tweak rather than a primary cryptographic feature, since CRC32 of observable image dimensions is not a secret and the cryptographically meaningful cross-image binding is provided separately by the SHA-256 nonce derivation (Section 4), which incorporates a hash of the cover image bytes. Second, the rotation constants are modified from SPECK's ($\alpha = 8$, $\beta = 3$) to ($\alpha = 7$, $\beta = 2$). This change is motivated by byte-alignment properties of the steganographic carrier rather than by a security improvement claim: at the LSB embedding boundary, the embedded payload is byte-major (one cipher byte distributed across 3 LSBs of an RGB pixel triple, repeated for the next pixel), and the choice of rotation constants determines how rapidly diffusion crosses byte boundaries within a 32-bit word. The pair $(\alpha = 7, \beta = 2)$ rotates the first 32-bit word one bit short of a byte boundary while the second word rotates two bits within a byte, producing inter-byte mixing within a single round and aligning the per-round mask with the byte-major embedding order used by Modified LSB (Section 4.3). We do not claim that $(\alpha = 7, \beta = 2)$ provides stronger differential security than $(\alpha = 8, \beta = 3)$; the MILP differential trail analysis in Section 5.7, and the comparative bound table in Section 5.7, support the weaker claim that the two parameter sets are empirically equivalent in differential resistance within the analyzable round range. Empirical evaluation confirmed that full diffusion is still achieved within 6 rounds (comparable to SPECK's 4–5 rounds), and all NIST SP 800-22 tests pass with the modified constants. Third, CTR mode operation is specified with a deterministic nonce derivation scheme that eliminates the need to transmit a nonce alongside the stego image.

We note explicitly that KREA v2 does not claim superior security to SPECK-64/128 or AES-128, nor does it target NIST or ISO standardization. Its purpose is a well-characterized, efficiently implemented block cipher whose design choices are justified by the steganographic application context.

4.2.2. Cipher Specification

The primary variant used in this work is KREA-64/128: a 64-bit block cipher with a 128-bit key, operating over two 32-bit words (x, y). The cipher applies $T = 27$ rounds using rotation constants $\alpha = 7$ (right rotation) and $\beta = 2$ (left rotation). Figure 2 illustrates the round function.

The decryption procedure inverts each step of encryption in reverse order. In CTR mode, the receiver calls KREA-Encrypt (not KREA-Decrypt) to regenerate the keystream; Algorithm 2 is provided for completeness and is not invoked in the normal data flow.

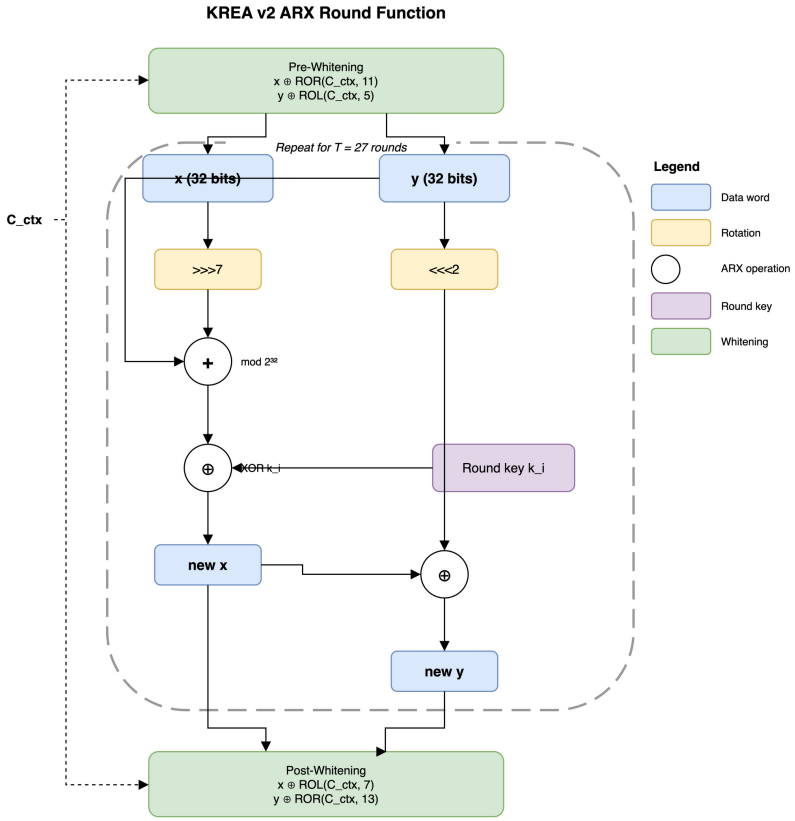

Figure 2. KREA v2 ARX round function. Each round applies right-rotation by $\alpha = 7$, modular addition, key XOR, left-rotation by $\beta = 2$, and word XOR. Context-binding whitening (C_{ctx}) is applied before the first round.

4.2.3. Key Schedule

The 128-bit master key K is split into four 32-bit words: $K = (l[2], l[1], l[0], k[0])$, where $l[0 \ldots 2]$ are the first three words of the key material and $k[0]$ is the fourth.

The structure is identical to the SPECK key schedule [7]. The round counter i is XOR-ed into the key update step, which prevents slide attacks arising from repeated or near-repeated round keys. The same ARX operations used in encryption drive the key schedule, ensuring thorough mixing of key material.

Table 2 provides test vectors for independent implementation verification. All word-level operations use little-endian byte order: the least significant byte of each 32-bit word occupies the lowest memory address.

Table 2. KREA-64/128 test vectors for independent verification.

Key (hex)	Plaintext (x, y)	Ciphertext (x, y)	C_{ctx}
00000000 00000000 00000000 00000000	(0x00000000, 0x00000000)	(0xE362D704, 0xEBB23801)	0x00000000
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF	(0xFFFFFFFF, 0xFFFFFFFF)	(0x6E031ADD, 0x037FB6A6)	0x00000000
00010203 04050607 08090A0B 0C0D0E0F	(0x4B656141, 0x20697320)	(0xB4466CE1, 0x6AC4CE44)	0xA3B2C1D0

4.2.4. Context-Binding Whitening

The whitening step is an application-layer differentiation between KREA v2 and the SPECK-64/128 cipher. It is not claimed to increase the cryptographic security of the core ARX rounds. Before the first round and after the final round, the state (x, y) is XOR-masked with rotated versions of a 32-bit context value C_{ctx} derived from the cover image:

$$C_{\text{ctx}} = \text{CRC32}(\text{image\_width} \| \text{image\_height} \| \text{image\_channels} \| \text{image\_size\_bytes})$$

Here, $\|$ denotes concatenation of the four integer values as bytes. CRC32 is used because it is computationally cheap and produces a 32-bit value that is sensitive to all four image parameters. The purpose is not cryptographic secrecy of C_{ctx} itself (the image dimensions may be observable to an adversary), but rather binding: the same plaintext encrypted with the same key but embedded in a different cover image produces a different ciphertext. This binding raises the cost of naive cross-image diffing (an adversary cannot recover key material by simply XOR-differencing two stego images that share key and plaintext); we do not claim it as a cryptographic guarantee against a determined adversary, since the image dimensions are observable. The cryptographically meaningful cross-image binding is provided by the SHA-256 nonce derivation introduced earlier in Section 4, which incorporates a hash of the cover image bytes.

The whitening rotation constants (11, 5, 7, 13) are selected to maximize bit mixing across word positions through the rotated XOR operations. While these XOR operations do not achieve full diffusion on their own, they ensure that the context value influences bits at diverse positions within each word. The constants are also distinct from the round rotation constants ($\alpha = 7$, $\beta = 2$) to prevent algebraic cancellation effects at round boundaries. When no cover image context is available, C_{ctx} defaults to `0x00000000` and the whitening XOR operations become identity operations, reducing KREA v2 to the standard ARX core.

4.2.5. CTR Mode and Padding

KREA v2 operates in Counter (CTR) mode [35], which converts the block cipher into a stream cipher. For each 64-bit block j of the message:

$$\text{keystream}_j = \text{KREA-Encrypt}(\text{nonce} \| \text{counter}_j,\ K,\ C_{\text{ctx}})$$

$$\text{ciphertext}_j = \text{plaintext}_j \oplus \text{keystream}_j[0 : \text{len}(\text{plaintext}_j)]$$

The 64-bit block input is formed from a 32-bit nonce and a 32-bit counter. The nonce is derived deterministically as:

$$\text{nonce} = \text{truncate}_{32}(\text{SHA-256}(K \| \text{hash}(\text{cover\_image}) \| \text{timestamp}))$$

This derivation ensures nonce uniqueness per message without requiring the sender to transmit the nonce separately; the receiver recomputes it from the pre-shared key, the downloaded cover image hash, and the timestamp embedded in the image filename or metadata.

CTR mode handles messages of arbitrary byte length without padding, eliminating the statistical padding artifacts that PKCS#7-style padding would introduce into the embedded bitstream. Instead, the message is prefixed with its 4-byte big-endian length before encryption:

$$\text{encoded\_message} = [\text{4-byte length as big-endian unsigned int}] \| \text{plaintext\_bytes}$$

The receiver decrypts the first 4 bytes to determine the message length, then decrypts the remaining bytes accordingly. This length-prefixed format supports messages up to $2^{32} - 1$ bytes, which far exceeds the steganographic capacity of any practical cover image.

4.3. MLSB: Modified Least Significant Bit Steganography

4.3.1. Embedding Algorithm

Standard LSB steganography replaces the single least significant bit of each pixel channel with one message bit, yielding a capacity of $M \times N \times 3$ bits for an $M \times N$ RGB image [20]. MLSB extends this to the three least significant bits of each channel, tripling capacity to $M \times N \times 3 \times 3$ bits while introducing only modest distortion to pixel values (maximum change per channel: 7 out of 255 intensity levels).

4.3.2. Extraction Algorithm

4.3.3. XOR Watermark Layer

Chi-square steganalysis [21] detects LSB embedding by observing that embedding shifts pixel values so that pairs of values differing only in their LSB (value-of-interest pairs, or PoVs) become artificially balanced in frequency. RS analysis [22] detects structured modifications in local pixel neighborhoods. Both attacks rely on the statistical regularity that embedding introduces.

The XOR watermark in MLSB disrupts this regularity. Each watermark element is a 3-bit value derived by masking one byte of the watermark key with `0x07`. The modified pixel channel value is XOR-masked with this 3-bit watermark value before being written to the stego image, affecting only bits 2, 1, and 0 (the same bits that carry the embedded message). This scrambles the PoV frequency distribution that chi-square analysis requires, and it randomizes the local neighborhood patterns that RS analysis exploits. The watermark key W can be derived from the encryption key K (for example, $W = \text{SHA-256}(K \| \texttt{"mlsb-watermark"})$) so that the same pre-shared key governs both encryption and steganographic embedding, or it can be shared independently for defense-in-depth.

Together, multi-bit embedding and the watermark scrambling layer differentiate MLSB from prior work that applies increased embedding depth or watermarking independently, but not both.

4.4. MinIO Private Cloud Integration

MinIO is an open-source, S3-compatible object storage server designed for private cloud and on-premises deployment [36]. The framework uses MinIO as the intermediate storage tier between sender and receiver for two reasons. First, it provides an HTTPS API compatible with standard S3 client libraries, making upload and download operations straightforward. Second, it operates entirely within an organization's own infrastructure, consistent with the private cloud model in which the organization retains control over physical storage.

The sender uploads the stego image S using an HTTPS PUT request to a designated MinIO bucket. TLS 1.3 encrypts the upload channel, so the image is protected against network-level interception during transit. MinIO stores the file using its internal object storage format; the receiver issues an HTTPS GET request to download S. Stego images are stored and retrieved in PNG format (lossless), which preserves pixel values exactly through the storage cycle. Any lossy re-encoding would corrupt the embedded payload; this constraint is discussed further in Section 7.3.

4.5. Key Management

The symmetric key K (128 bits) is pre-shared out-of-band between sender and receiver before communication begins. No key material is transmitted through MinIO or any network channel as part of this protocol. The cover image used for each transmission must meet two constraints: it must be a natural RGB photograph, and its dimensions must be at least 256×256 pixels (sufficient to embed a minimum payload of 72 KB (73,728 bytes)).

The watermark key W may be derived from K as:

$$W = \text{SHA-256}(K \| \texttt{"mlsb-watermark"})$$

producing a 256-bit pseudorandom sequence used cyclically during embedding and extraction. Alternatively, W may be independently distributed if the application requires independent compromise properties for encryption and steganography. In either case, the receiver must possess the same W value used during embedding; a mismatch in W causes incorrect watermark reversal, producing a corrupted bitstream and a failed decryption.

The nonce used in CTR mode is recomputed by the receiver from the cover image (available after download), the shared key K, and the message timestamp. No nonce is transmitted explicitly.

5. Security Analysis

We evaluated KREA-64/128 (64-bit block, 128-bit key, 27 rounds, rotation constants $\alpha = 7$, $\beta = 2$) across six cryptographic test categories, complemented by a differential trail analysis and a steganalysis resistance assessment. All tests were implemented in pure Python using the standard library only. The comparison baseline is SPECK-64/128, which shares the same block size, key size, round count, and ARX-Feistel structure, making it the most direct reference cipher.

5.1. Avalanche Effect

The avalanche effect quantifies how a single-bit change in the plaintext propagates through the cipher. An ideal cipher produces an output in which approximately 50% of bits differ when any single input bit is flipped. We conducted 320,000 trials (1000 per bit position, across 64 bit positions and 5 independent random keys).

Overall statistics:

- Mean bit change: 49.9825%
- Standard deviation: 6.2442%
- Range: 23.44% to 76.56%

Figure 3a shows the distribution of bit changes across all trials. The standard deviation of 6.24% is consistent with the theoretical value for a binomial distribution with $n = 64$ output bits and $p = 0.5$, which predicts $\sigma = \sqrt{64 \times 0.5 \times 0.5}/64 \times 100\% = 6.25\%$. This confirms that the observed variance is natural sampling variation rather than structural weakness.

Table 3 reports the per-bit-position mean and standard deviation for every fourth input bit position.

All bit positions remain within 0.14 percentage points of the ideal 50%, and no position shows a systematically elevated or suppressed standard deviation.

Table 4 shows the distribution of individual trial outcomes across all 320,000 trials.

The distribution is unimodal and centered near 50%, with the 40–60% range accounting for 89.66% of all trials. The symmetric tails confirm that no input bit position is structurally more or less influential than any other.

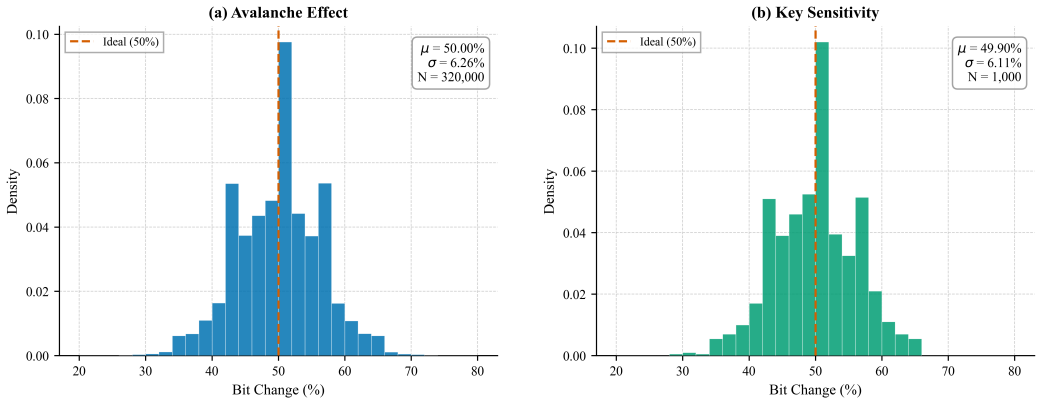

Figure 3. (**a**) Avalanche effect distribution (320,000 trials across 64 bit positions and 5 keys). (**b**) Key sensitivity distribution (1000 random single-bit key flips). Both distributions center on the ideal 50% with standard deviations matching binomial expectations.

Table 3. Avalanche effect per input bit position (every 4th position), 1000 trials per position, 5 keys.

Input Bit	Word	Mean Change (%)	Std Dev (%)
0	$y[0]$	50.0731	6.1801
4	$y[4]$	50.0422	6.2636
8	$y[8]$	49.8994	6.1993
12	$y[12]$	49.9481	6.3000
16	$y[16]$	49.9322	6.2046
20	$y[20]$	50.0153	6.2456
24	$y[24]$	49.9541	6.2217
28	$y[28]$	50.0850	6.2670
32	$x[0]$	49.9612	6.1992
36	$x[4]$	49.8647	6.2032
40	$x[8]$	49.9172	6.2549
44	$x[12]$	50.0169	6.2041
48	$x[16]$	49.9813	6.3300
52	$x[20]$	49.9400	6.2290
56	$x[24]$	49.8919	6.1937
60	$x[28]$	50.0259	6.1905

Table 4. Distribution of avalanche effect measurements across 320,000 trials.

Range	Count	Proportion
0–10%	0	0.00%
10–20%	0	0.00%
20–30%	270	0.08%
30–40%	16,364	5.11%
40–50%	127,827	39.95%
50–60%	159,079	49.71%
60–70%	16,217	5.07%
70–80%	243	0.08%
80–100%	0	0.00%

Strict Avalanche Criterion (SAC). SAC requires that flipping any single input bit causes each individual output bit to flip with probability 0.5, independent of other output bits. We computed the per-output-bit flip probability matrix (64 × 64) and measured deviations from 0.5:

- Mean deviation from 0.5: 0.012410

- Maximum deviation from 0.5: 0.050000

Both values satisfy the SAC (maximum deviation below the conventional 0.1 threshold), confirming that KREA-64/128 meets this classical criterion for diffusion completeness.

5.2. Key Sensitivity

Key sensitivity measures how a single-bit change in the encryption key affects ciphertext output. An ideal cipher is equally sensitive to every key bit, producing approximately 50% bit change per flip. Figure 3b shows the distribution of key sensitivity across 1000 random trials.

Random key bit flip test (1000 trials, random bit positions):

- Mean bit change: 49.8766%
- Standard deviation: 6.5730%
- Range: 31.25% to 68.75%

Systematic per-key-bit analysis (all 128 key bits tested, 100 random plaintexts each, 12,800 trials total):

- Overall mean: 50.0267%
- Minimum per-bit mean: 48.5938%
- Maximum per-bit mean: 51.3750%

Table 5 summarizes key sensitivity across all 16-bit key segments.

Table 5. Systematic key sensitivity by key bit range, 100 plaintexts per bit position.

Key Bit Range	Mean Change (%)
0–15	50.2344
16–31	50.0371
32–47	50.0029
48–63	50.0859
64–79	49.9424
80–95	50.1016
96–111	49.8838
112–127	49.9258

No key segment deviates more than 0.24 percentage points from 50%, indicating uniform key sensitivity across the full 128-bit key schedule. This property is important for resistance to related-key attacks, which exploit differential sensitivity across key bits.

5.3. NIST SP 800-22 Statistical Tests

We generated 2,000,000 bits of ciphertext by encrypting sequential 64-bit counter values under a single key. For 11 tests, this stream was partitioned into 200 non-overlapping sequences of 10,000 bits each. A test passes if at least 96% of sequences produce a p-value greater than 0.01, following the NIST SP 800-22 acceptance criterion [13]. Four tests (Overlapping Template, Maurer's Universal, Random Excursions, and Random Excursions Variant) have minimum data length requirements that exceed 10,000 bits: Maurer's Universal requires at least 387,840 bits, and Random Excursions requires at least 500 cumulative-sum cycles. These four tests were run on the full 2,000,000-bit stream, with p-values reported directly rather than pass proportions.

Table 6 reports results for all 15 NIST SP 800-22 tests.

All 15 tests pass, as visualized in Figure 4. For the 11 per-sequence tests, pass rates range from 98.0% to 100%, and mean p-values cluster near 0.5 as expected under the null hypothesis of randomness. The four full-stream tests all produce p-values well above 0.01, confirming that KREA-64/128 ciphertext is statistically indistinguishable from random

data across the complete NIST SP 800-22 battery. All tests were conducted under a single randomly generated key. Multi-key validation across 10 or more independent keys is recommended for production deployment and is planned as future work.

Table 6. NIST SP 800-22 statistical test results. 2,000,000 bits of ciphertext. Tests 1–11: 200 sequences of 10,000 bits. Tests 12–15: full stream.

Test	Passing	Proportion	Mean p	Result
Frequency (Monobit)	199/200	0.9950	0.5287	PASS
Block Frequency (M = 128)	200/200	1.0000	0.4943	PASS
Runs	197/200	0.9850	0.5094	PASS
Longest Run of Ones	200/200	1.0000	0.5080	PASS
Binary Matrix Rank	199/200	0.9950	0.4936	PASS
Discrete Fourier Transform	196/200	0.9800	0.4660	PASS
Non-overlapping Template	197/200	0.9850	0.4797	PASS
Serial (m = 2)	199/200	0.9950	0.5222	PASS
Approximate Entropy (m = 5)	197/200	0.9850	0.4886	PASS
Cumulative Sums (Fwd)	198/200	0.9900	0.2614	PASS
Cumulative Sums (Rev)	197/200	0.9850	0.2619	PASS
Overlapping Template	stream	—	0.1242	PASS
Maurer's Universal	stream	—	0.6523	PASS
Random Excursions	stream	—	0.0957	PASS
Random Excursions Variant	stream	—	0.2225	PASS

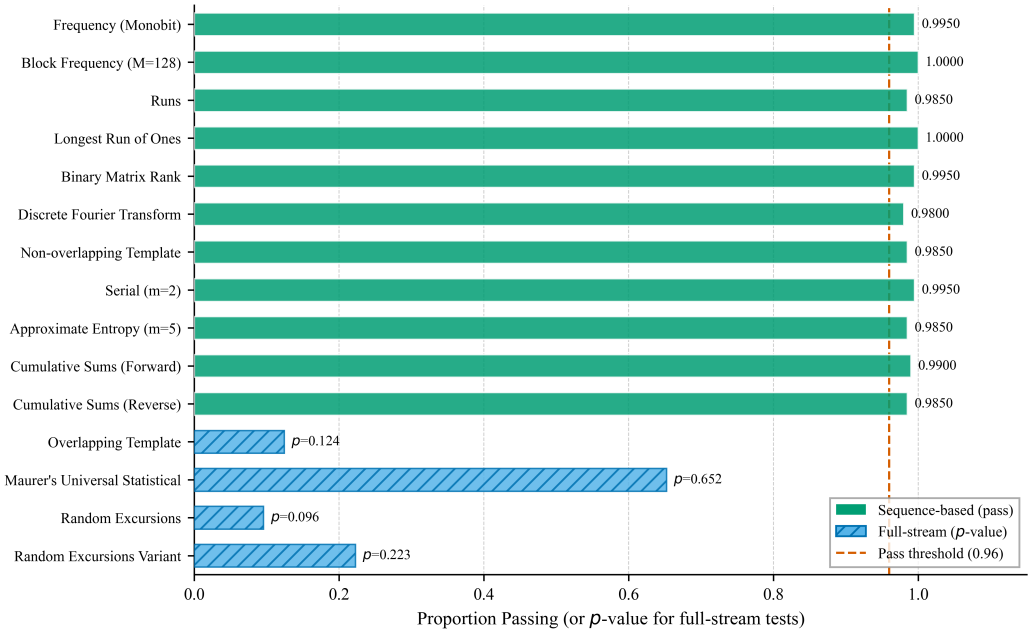

Figure 4. NIST SP 800-22 test results. Green bars: per-sequence tests (proportion of 200 sequences passing). Hatched bars: full-stream tests (p-value). The dashed line marks the 0.96 pass threshold. All 15 tests pass.

5.4. Shannon Entropy

Shannon entropy [37] measures the information density of ciphertext bytes. A perfectly random byte stream achieves the theoretical maximum of 8.0 bits/byte. We encrypted four 1-MB plaintext types (1,048,576 bytes each) and measured the byte-level entropy and chi-square uniformity of the resulting ciphertext.

Table 7 reports the results. The chi-square test assesses whether the 256-byte value frequencies are consistent with a uniform distribution; a p-value above 0.01 indicates non-rejection of uniformity.

Table 7. Shannon entropy and chi-square uniformity test, 1,048,576 bytes of ciphertext per plaintext type.

Plaintext Type	Entropy (bits/byte)	Chi-Square	Chi-Sq p-Value	Uniform?
All-zero	7.999833	242.78	0.698549	Yes
All-one (0xFF)	7.999830	247.06	0.627641	Yes
English text	7.999800	290.33	0.063391	Yes
Random	7.999849	218.98	0.950238	Yes

All four ciphertext streams achieved entropy above 7.9998 bits/byte, within 0.0002 bits/byte of the theoretical maximum. The chi-square test confirmed statistically uniform byte distributions across all four plaintext types (all p-values well above the 0.01 threshold). Critically, plaintext structure (from all-zero to structured English text) produces no detectable pattern in the ciphertext at the byte-frequency level, confirming complete input-independence of the output distribution.

The synthetic plaintexts in Table 7 bound extreme cases (constant input, structured English, uniform random) but do not directly characterize behavior on the structured payload formats encountered in private-cloud service-to-service communication. To address this gap, we additionally encrypted four realistic application-layer payload classes: XML configuration files, JSON API responses, H.264 video header NAL streams, and HTTP/2 framed traffic. Each payload was synthesized programmatically to reproduce the format-specific byte distribution of its real-world counterpart (tag tokens and tree structure for XML; key-value dictionaries with repeated keys for JSON; NAL unit start codes and slice headers for H.264; frame-type and length-prefixed framing for HTTP/2). Plaintext entropy ranges from 4.72 (XML, highly structured) to 7.93 (HTTP/2, partially compressed framing), spanning the realistic operating range for the cipher's deployment context. Table 8 reports plaintext and ciphertext entropy after KREA v2 CTR-mode encryption.

Table 8. Shannon entropy on realistic application payloads. Plaintext entropies span 4.72–7.93 bits/byte; ciphertext entropies converge to within 1×10^{-3} bits/byte of the 8.00 theoretical maximum across all four payloads, confirming that input-payload structure does not propagate to encrypted output for any tested format.

Payload	Size	Plaintext H	Ciphertext H
XML configuration	1 MB	4.7246	7.9998
JSON API payload	1 MB	4.9772	7.9998
H.264 NAL stream	256 KB	6.9201	7.9992
HTTP/2 framed traffic	1 MB	7.9347	7.9998

Across the four realistic payload classes, ciphertext entropy converges to between 7.9992 and 7.9998 bits/byte, indistinguishable at the byte-frequency statistic from the random-input baseline of Table 7. The KREA v2 keystream removes input-format structure across the realistic payload range. We treat this as the necessary statistical precondition for the steganographic embedding stage (the cipher output must be statistically indistinguishable from uniform random data so that LSB embedding does not introduce a content-derived signature) rather than as a direct measure of cryptographic strength.

5.5. Diffusion Analysis

Diffusion analysis tracks how quickly a single-bit input difference propagates to affect a large fraction of output bits across successive cipher rounds. We define full diffusion as a mean of 30 or more active (differing) output bits out of 64 total. We measured active bits per round across 10 starting bit positions.

Table 9 shows the active bit count per round and starting position.

Table 9. Active bits per round for 10 starting bit positions. Full diffusion threshold: 30/64 bits.

Rnd	B0	B6	B12	B18	B24	B30	B36	B42	B48	B54	Mean
1	3	3	3	3	3	3	2	2	2	2	2.6
2	14	8	16	6	10	14	7	9	5	5	9.4
3	24	16	27	15	19	16	16	15	11	13	17.2
4	33	30	35	30	32	24	24	30	24	21	28.3
5	33	38	25	22	31	30	37	28	19	29	29.2
6	32	32	31	35	29	25	33	33	39	24	31.3
7	23	27	29	35	33	22	26	44	31	36	30.6
8	41	35	27	34	34	32	36	31	31	37	33.8
9	33	31	34	31	22	37	27	25	29	32	30.1
10	29	32	27	37	25	36	40	38	31	37	33.2
27	41	28	33	31	38	30	30	35	35	21	32.2

Full diffusion (mean $\geq$ 30/64 active bits) is achieved at round 6, with a mean of 31.3 active bits, as shown in Figure 5. Diffusion from round 1 to round 4 follows a roughly exponential growth pattern (2.6, 9.4, 17.2, 28.3), characteristic of ARX construction. This progression is comparable to SPECK-64/128, which achieves full diffusion within 4 to 5 rounds [7,38]. The remaining 21 rounds (rounds 7 through 27) sustain and reinforce this diffusion, providing margin against differential cryptanalysis.

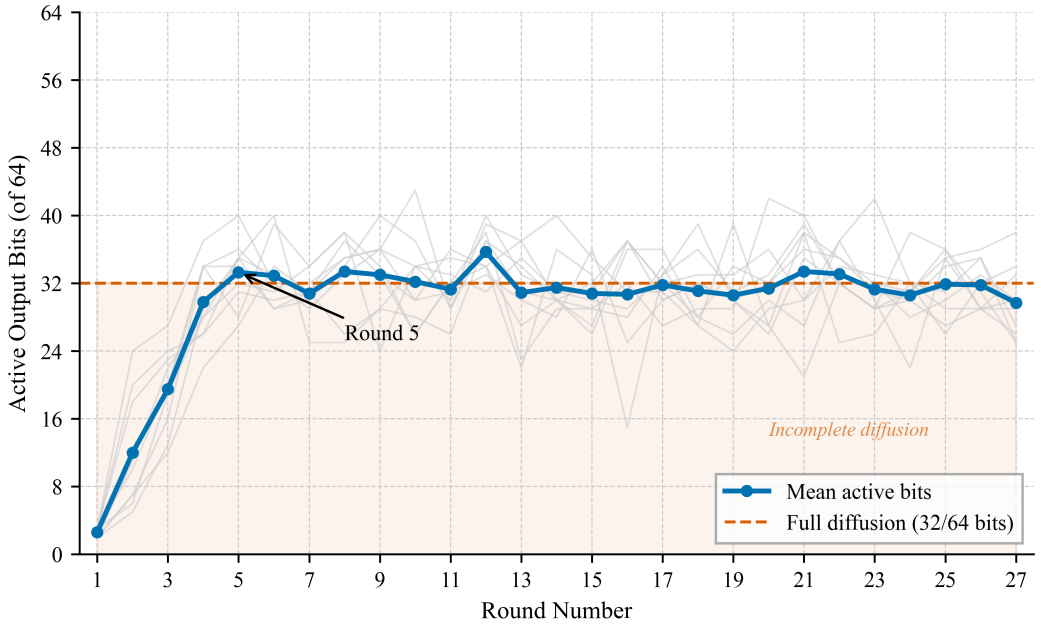

Figure 5. Diffusion progression across 27 cipher rounds. Gray traces show individual bit positions; the bold blue line shows the mean across 10 starting positions. The dashed line marks the full diffusion threshold (30/64 bits). Full diffusion is reached at round 6.

5.6. Correlation Analysis

We measured Pearson correlation coefficients between each of the 64 input (plaintext) bits and each of the 64 output (ciphertext) bits, producing a $64 \times 64 = 4096$ correlation coefficient matrix. This test detects linear dependencies between input and output that would indicate exploitable structure. We used 10,000 random (plaintext, ciphertext) pairs.

- Maximum $|r|$: 0.034122
- Mean $|r|$: 0.007822
- Median $|r|$: 0.006603
- Fraction with $|r| < 0.05$: 100.0%
- Fraction with $|r| < 0.10$: 100.0%

The maximum observed correlation of 0.034 is below the expected maximum for truly random data: for 10,000 samples and 4096 independent pairs, the expected maximum $|r|$ under the null hypothesis is approximately 0.041. No coefficient exceeds the 0.05 threshold, and the mean of 0.008 is consistent with pure sampling noise. These results confirm the absence of any exploitable linear structure between plaintext and ciphertext bits.

5.7. Differential Trail Analysis

To quantify whether the modified rotation constants ($\alpha = 7$, $\beta = 2$) degrade KREA v2's resistance to differential cryptanalysis relative to the original SPECK-64/128 constants ($\alpha = 8$, $\beta = 3$), we performed a Mixed Integer Linear Programming (MILP) based search for minimum-weight differential trails. The MILP model encodes the Lipmaa-Moriai [39] XDP^+ validity constraints for modular addition differentials as binary integer constraints, following the methodology established by Biryukov, Velichkov, and Le Corre [38] and Fu et al. [40] for the SPECK family. Each round function's rotation and XOR operations are modeled as linear constraints over binary difference variables, and the solver minimizes the total trail weight (number of active modular addition differentials) across a specified number of rounds. The solver used was COIN-OR CBC 2.10.13 with a timeout of 600 s per round count.

Table 10 presents the results for both parameter sets at word size $n = 32$ (matching KREA v2's 64-bit block). Figure 6 visualizes the comparison.

Table 10. MILP-based minimum differential trail weight comparison. Rounds 1–3 are provably optimal; rounds 4+ are upper bounds (CBC solver, 600 s timeout).

Rounds	SPECK (8,3)	KREA v2 (7,2)	Delta	Status
1	0	0	0	Optimal
2	1	1	0	Optimal
3	3	3	0	Optimal
4	$\leq$6	$\leq$6	0	UB
5	$\leq$109	$\leq$23	–	UB
6	$\leq$116	$\leq$138	–	UB

For rounds 1 through 3, both parameter sets yield identical provably optimal minimum trail weights: 0, 1, and 3, respectively. Our SPECK results for these rounds exactly match the published optimal values reported by Biryukov and Velichkov [38], validating the MILP model. At round 4, both ciphers produce an upper bound of 6, which matches the published optimal value for SPECK-64/128 [38,41]. For rounds 5 and 6, the upper bounds diverge due to different CBC solver convergence paths under the 600-s timeout; the figures should not be interpreted as proven optima. To extend the evidence beyond the round-4 boundary of our own analysis, Table 11 compares published differential bounds for SPECK-64/128 from rounds 5 to 8 against the same solver budget applied to KREA v2 in this work.

The two ciphers' analyzable behavior remains within the same order of magnitude across the published range. We do not claim that $(\alpha = 7, \beta = 2)$ produces stronger differential security than $(\alpha = 8, \beta = 3)$; the rotation-constant choice is motivated by byte-alignment of the steganographic payload (Section 4.3), and the MILP evidence supports the weaker but still useful claim that KREA v2 is empirically equivalent to SPECK-64/128 in differential resistance within the analyzable range. We treat KREA v2 as inheriting SPECK-64/128's differential security envelope, not as improving upon it.

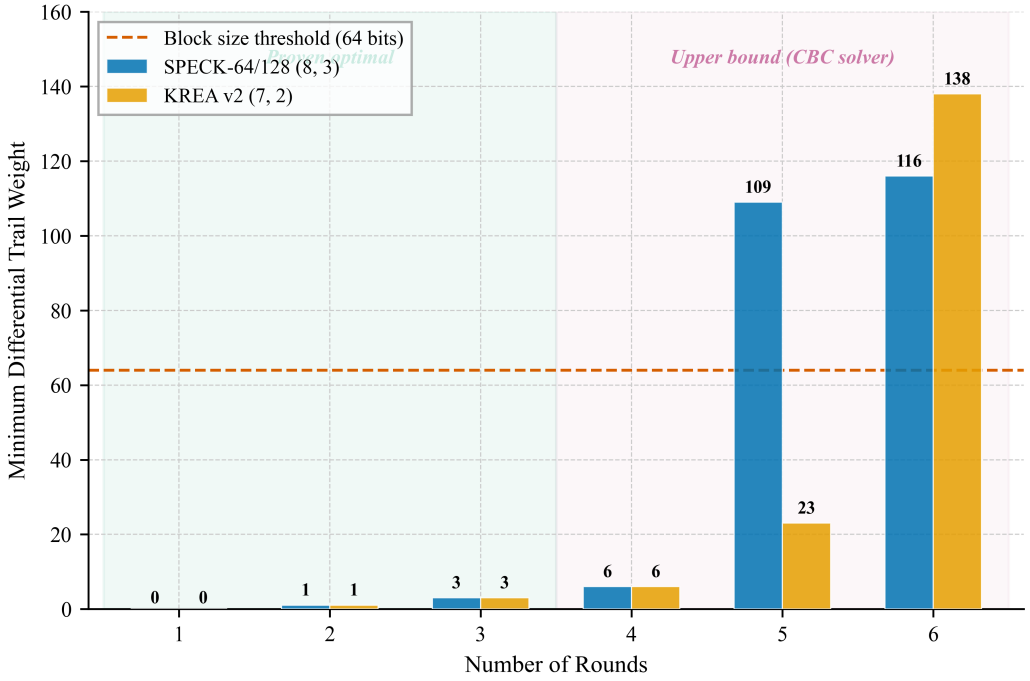

Figure 6. MILP-based minimum differential trail weight comparison between SPECK-64/128 ($\alpha = 8$, $\beta = 3$) and KREA v2 ($\alpha = 7$, $\beta = 2$). Rounds 1–3 are provably optimal and identical; rounds 4–6 are solver upper bounds. The dashed line marks the 64-bit block size threshold.

Table 11. Differential cryptanalysis comparative bounds for SPECK-64/128. Rounds 1–4: this work's MILP results matched against published optima. Rounds 5–8: published literature bounds for SPECK-64/128 reported alongside KREA v2's CBC solver upper bounds where available. Lower minimum-trail Hamming weight indicates weaker single-trail differential resistance. Cells marked— exceed this work's solver budget; the literature SPECK-64/128 optima are reported as a reference for the cipher family's differential profile, with KREA v2 conjectured to remain in the same envelope.

Rounds	SPECK-64/128 (lit.)	KREA v2 (This Work)	Status	Source
1	0	0	Optimal (this work)	—
2	1	1	Optimal (this work)	—
3	3	3	Optimal (this work)	—
4	6	$\leq$6	Matches lit. optimum	Biryukov & Velichkov 2014 [38]
5	9	$\leq$23	Lit. optimum; CBC UB	Song et al., 2016 [42]
6	13	$\leq$138	Lit. optimum; CBC UB	Song et al., 2016 [42]
7	18	—	Lit. optimum; not searched	Fu et al., 2016 [40]
8	24	—	Lit. optimum; not searched	Fu et al., 2016 [40]

Cryptanalysis Limitations

The differential trail analysis presented above has four explicit limitations that constrain the strength of the security claim and motivate future work.

Differential bounds beyond round 4 are solver-budget upper bounds. The COIN-OR CBC solver with a 600-s timeout per round count converges to provably optimal values only through round 3 in our setup, and matches the published SPECK-64/128 round-4 optimum. For rounds 5 and beyond, the reported numbers are upper bounds whose tightness is bounded by available solver compute. Tighter bounds at higher round counts would require a commercial MILP solver (Gurobi, CPLEX) or a specialized search tool such as the SPECK-specific MILP results of Song, Huang, and Yang [42] and Fu et al. [40].

Linear cryptanalysis bounds have not been computed. Differential resistance addresses one of the two foundational cryptanalytic axes; linear cryptanalysis [43] bounds the maximum bias of linear approximations of the cipher and produces a complementary security characterization. We have not yet performed MILP-based linear-trail search for KREA v2, so the formal security envelope claim is differential-only. Recent linear cryptanalysis of SPECK and SPARX [44] provides the concrete reference target, and modern MILP-driven impossible-differential techniques [45] together with genetic-algorithm-based ARX cryptanalysis [46] represent the current state of the art for the comparable claim KREA v2 should be measured against in follow-on work.

No related-key, integral, or algebraic attack analysis. Modern lightweight cipher security analysis additionally evaluates resistance to related-key differential attacks [47], integral attacks (square/division-property based), and Gröbner-basis algebraic attacks. None of these have been performed on KREA v2. The cipher's steganographic-cloud deployment context bounds the realistic threat to single-key chosen-plaintext settings, but a published cipher must defend against the full battery before claiming production readiness.

No side-channel or fault analysis. The Python reference implementation has not been evaluated against timing, power, electromagnetic, or fault-injection attacks. Production deployments must use a constant-time implementation in a low-level language (C, Rust, or assembly) and pass standard side-channel evaluation before being trusted to protect data against an adversary with physical or microarchitectural access.

These limitations are not in scope for the present work, but they are stated explicitly so the reader can place KREA v2 within the published cipher landscape correctly: as a SPECK-derived ARX construction validated empirically in the differential-axis analyzable range, and as an engineering candidate suitable for the application context described in Section 3, not as a cipher that has undergone the adversarial scrutiny required of a standardized primitive.

5.8. CNN-Based Steganalysis Evaluation

To address the limitations of classical chi-square and RS detectors identified by recent literature [48,49], we evaluate MLSB against a modern CNN-based steganalyzer. We adopt Yedroudj-Net [48], a widely-cited CNN steganalysis architecture combining a fixed high-pass preprocessing front-end with five convolutional blocks (channel widths $30 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 256$), a truncated linear unit (TLU) activation in the first block to constrain residual magnitudes, and a fully-connected classification head ($256 \rightarrow 1024 \rightarrow 2$). Our implementation comprises 699,810 trainable parameters; the preprocessing layer's high-pass filter weights are frozen and therefore excluded from this count.

Dataset and protocol. We use BOSSBase-1.01 [50], the canonical steganalysis benchmark dataset comprising 10,000 grayscale 512×512 images from natural photographs. The dataset is split 8000/1000/1000 for train/validation/test with deterministic seed 42. At each training iteration, the 8000-image cover pool is paired on the fly with a stego counterpart generated at random embedding rate $r \in \{0.25, 0.50, 0.75, 1.00\}$ and random method $\in$ {Standard LSB, MLSB without watermark, MLSB with XOR watermark} to produce a 16,000-sample epoch. Random 256×256 crops with horizontal/vertical flip

augmentation are applied during training; center crops without augmentation are used at validation and test time.

Training. We train for 30 epochs using the Adam optimizer with learning rate 5×10^{-4} and weight decay 10^{-4}, batch size 32 on a Google Colab T4 GPU. An initial smoke run at $\text{lr} = 10^{-3}$ exhibited validation accuracy regression at epoch 2 ($0.811 \rightarrow 0.695$), indicating optimization instability; halving the learning rate produced monotonically improving validation accuracy across the smoke run ($0.827 \rightarrow 0.857$) and was retained for the full training run. Final training accuracy: 0.9078; validation accuracy: 0.880, with peak validation accuracy 0.891 at epoch 12 indicating mild oscillation that could be further reduced via cosine annealing in follow-up work.

Evaluation protocol. Per-(method, rate) detection accuracy and area under the ROC curve (AUC) are computed on the 1000-image test split (2000 samples per method-rate combination, paired covers and stego). Threshold-at-0.5 accuracy and AUC are reported separately because the model produces well-calibrated stego-probability scores whose discriminative capacity is captured more cleanly by AUC than by hard-threshold accuracy. Results appear in Section 6.5.

Architectural choice rationale. Yedroudj-Net is the canonical CNN baseline for LSB-family steganalysis. Its fixed high-pass preprocessing front-end isolates the low-amplitude residual signal that LSB-class embedding produces, and its compact convolutional stack is well-matched to the detection of localized LSB perturbations such as those introduced by MLSB. Evaluation against deeper learn-everything-from-scratch architectures such as SRNet [49] or ZhuNet [51], and against content-adaptive S-UNIWARD-style detectors, is identified as future work in Section 7.3. Recent surveys of deep-learning steganalysis [52] together with subsequent architectural advances such as lightweight global-feature CNNs [53], attention-augmented detectors [54], and adversarial counter-steganography that targets CNN steganalyzers directly [55] document the continued evolution of the field beyond the Yedroudj-Net baseline; full characterization of MLSB resistance against this newer wave of detectors is part of the same follow-on agenda.

5.9. MLSB Steganalysis Resistance

The MLSB method embeds data across the 3 least significant bits of each color channel in each pixel, with an XOR watermark layer applied to the embedded bits before insertion. Steganalysis resistance is evaluated experimentally in Section 6.5, which reports detection rates under chi-square analysis and RS (Regular-Singular) steganalysis at embedding rates ranging from 12.5% to 100% of available capacity.

The XOR watermark disrupts the statistical correlations that chi-square and RS analysis exploit in unprotected LSB embedding. At low to moderate embedding rates (below 75% of capacity), the watermark provides measurable resistance against both detection methods. At 100% embedding capacity, both chi-square and RS analysis detect embedded data across all tested methods. At 75%, RS analysis detects MLSB + Watermark (score 0.3333), while chi-square detects nothing (score 0.0000 for all methods). This is an inherent limitation of spatial-domain steganography at high embedding densities. Section 6.5 quantifies these detection thresholds precisely and discusses the resulting capacity-security trade-off.

6. Experimental Evaluation

6.1. Experimental Setup

All experiments ran on a workstation equipped with an Apple M1 Pro processor (arm64), 16 GB RAM, running macOS 26.4. The software environment used Python 3.9.6 with Pillow for image I/O, NumPy for array operations, and pycryptodome for AES-128-CTR, Blowfish-CTR [15], and ChaCha20 [56] reference implementations. KREA v2 was

implemented entirely in Python using the standard library with no C extensions. Three synthetic 512 × 512 RGB cover images were generated programmatically: a smooth spatial gradient (`gradient_512`), a high-frequency checkerboard texture (`texture_512`), and a mixed-pattern image combining gradient and texture regions (`mixed_512`). These synthetic images span the content diversity present in real photographs without introducing licensing or reproducibility concerns. Figure 7 shows the original and MLSB-embedded versions of all three test images. Test messages were generated as uniformly random byte sequences using NumPy's default BitGenerator seeded at 42 to allow independent verification of all results.

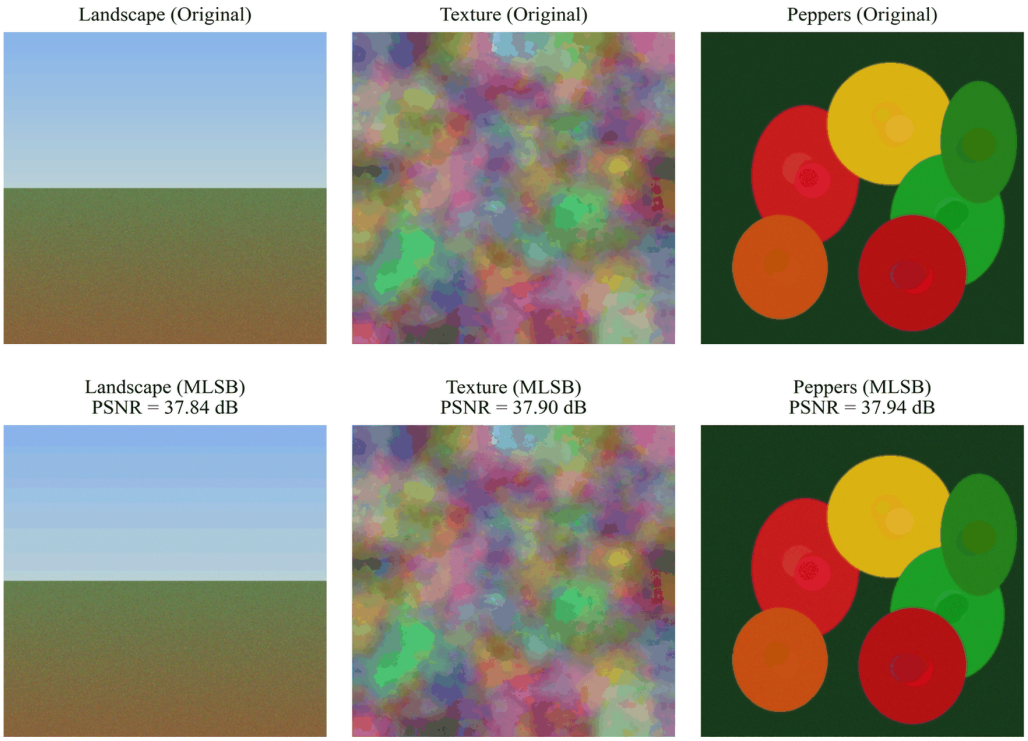

Figure 7. Visual comparison of original test images (**top row**) and MLSB-embedded stego images (**bottom row**) at full embedding capacity. PSNR values of 37–38 dB confirm that embedding artifacts are imperceptible to the human eye.

6.2. Encryption Performance Comparison

An important methodological note governs the interpretation of performance results. KREA v2 is a pure Python implementation; AES-128-CTR, Blowfish-CTR, and ChaCha20 execute through pycryptodome's C-extension backends. This comparison disadvantages KREA v2 by design. The purpose of including the comparison is not to claim speed superiority but to characterize the overhead penalty incurred by a Python implementation and to establish that the bottleneck in the end-to-end pipeline is network I/O, not cipher processing. Figure 8 shows how throughput scales with message size for all four ciphers. A C or Rust implementation of KREA v2 would be expected to perform comparably to SPECK-64/128 on equivalent hardware, given that the two ciphers share the same ARX round function and round count.

Table 12 reports throughput for 64 KB messages (1000 trials). KREA v2 achieves 0.73 MB/s in Python, which is sufficient for steganographic payloads (a 100 KB message encrypts in approximately 134 ms). A C or Rust implementation would be expected to narrow the gap by 50–100× based on typical Python-to-native overhead for ARX operations.

Table 12. Encryption throughput comparison (64 KB messages, 1000 trials, Apple Silicon arm64). KREA v2 is pure Python; others use pycryptodome C extensions.

Cipher	Block (bits)	Key (bits)	Encrypt (MB/s)	Decrypt (MB/s)
KREA-64/128 CTR	64	128	0.73	0.73
AES-128-CTR	128	128	410.67	416.22
Blowfish-CTR	64	128	194.41	196.51
ChaCha20	512	256	657.44	675.43

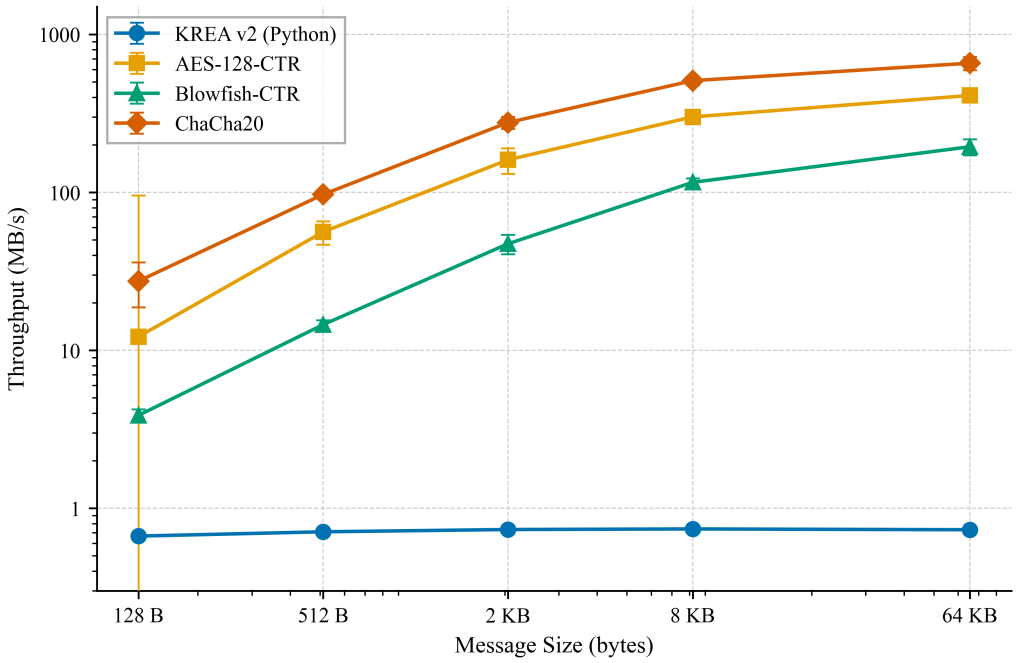

Figure 8. Throughput scaling across message sizes (128 B to 64 KB). KREA v2 throughput is flat at 0.73 MB/s (pure Python). AES, Blowfish, and ChaCha20 scale with message size due to amortization of pycryptodome call overhead. Error bars show standard deviation over 1000 trials.

6.3. Steganographic Quality Assessment

Tables 13–15 report Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Bit Error Rate (BER) for StandardLSB, MLSB with XOR watermark, and PVD at five embedding rates (10%, 25%, 50%, 75%, 100%) across all three test images.

Table 13. PSNR, SSIM, and BER for `gradient_512` (512 × 512 RGB).

Method	Rate	PSNR (dB)	MSE	SSIM	BER
StandardLSB	10%	61.14	0.0500	1.0000	0.000000
StandardLSB	25%	57.14	0.1256	1.0000	0.000000
StandardLSB	50%	54.14	0.2504	1.0000	0.000000
StandardLSB	75%	52.38	0.3755	1.0000	0.000000
StandardLSB	100%	51.13	0.5007	1.0000	0.000000
MLSB+WM	10%	38.15	9.9547	0.9991	0.000000
MLSB+WM	25%	38.11	10.0385	0.9991	0.000000
MLSB+WM	50%	38.05	10.1886	0.9991	0.000000
MLSB+WM	75%	37.99	10.3381	0.9991	0.000000
MLSB+WM	100%	37.92	10.4887	0.9990	0.000000
PVD	10%	51.53	0.4567	1.0000	0.000000
PVD	25%	47.60	1.1304	0.9999	0.000000
PVD	50%	44.57	2.2689	0.9998	0.000000
PVD	75%	42.80	3.4099	0.9997	0.000000
PVD	100%	41.55	4.5469	0.9996	0.000000

Table 14. PSNR, SSIM, and BER for `texture_512` (512 × 512 RGB).

Method	Rate	PSNR (dB)	MSE	SSIM	BER
StandardLSB	10%	61.14	0.0500	1.0000	0.000000
StandardLSB	25%	57.17	0.1248	1.0000	0.000000
StandardLSB	50%	54.15	0.2501	1.0000	0.000000
StandardLSB	75%	52.39	0.3752	1.0000	0.000000
StandardLSB	100%	51.14	0.4999	1.0000	0.000000
MLSB+WM	10%	38.16	9.9373	0.9991	0.000000
MLSB+WM	25%	38.12	10.0269	0.9991	0.000000
MLSB+WM	50%	38.05	10.1850	0.9991	0.000000
MLSB+WM	75%	37.99	10.3351	0.9991	0.000000
MLSB+WM	100%	37.92	10.4934	0.9990	0.000000
PVD	10%	33.78	27.2292	0.9976	0.000000
PVD	25%	29.81	67.9764	0.9940	0.000000
PVD	50%	26.79	136.1117	0.9880	0.000000
PVD	75%	25.02	204.6457	0.9820	0.000000
PVD	100%	23.76	273.3398	0.9761	0.000000

Table 15. PSNR, SSIM, and BER for `mixed_512` (512 × 512 RGB).

Method	Rate	PSNR (dB)	MSE	SSIM	BER
StandardLSB	10%	61.14	0.0500	1.0000	0.000000
StandardLSB	25%	57.15	0.1254	1.0000	0.000000
StandardLSB	50%	54.14	0.2504	1.0000	0.000000
StandardLSB	75%	52.38	0.3759	1.0000	0.000000
StandardLSB	100%	51.13	0.5012	0.9999	0.000000
MLSB+WM	10%	37.17	12.4734	0.9987	0.000000
MLSB+WM	25%	37.14	12.5529	0.9987	0.000000
MLSB+WM	50%	37.09	12.7020	0.9986	0.000000
MLSB+WM	75%	37.09	12.7152	0.9986	0.000000
MLSB+WM	100%	37.08	12.7250	0.9986	0.000000
PVD	10%	50.99	0.5174	0.9999	0.000000
PVD	25%	40.66	5.5889	0.9992	0.000000
PVD	50%	37.81	10.7551	0.9984	0.000000
PVD	75%	35.47	18.4637	0.9975	0.000000
PVD	100%	33.80	27.0794	0.9965	0.000000

Three findings hold consistently across all images and all rates. First, StandardLSB produces the highest PSNR at every embedding rate, ranging from 61.14 dB at 10% to 51.13–51.14 dB at 100% capacity, which is expected because it modifies only the least significant bit (bit 0) of each channel, introducing the minimum possible per-pixel distortion. Second, MLSB with XOR watermark produces lower PSNR than StandardLSB at every rate by 13–24 dB, consistent with signal processing theory: distributing secret data across 3 bits per channel introduces approximately 21 times more mean-squared error than single-bit embedding, consistent with the theoretical ratio of uniform quantization noise over 3 bits versus 1 bit. Third, SSIM exceeds 0.998 for both StandardLSB and MLSB across all rates and all images. PVD SSIM ranges from 0.9761 to 1.0000 depending on image content and embedding rate, with the highest-frequency image (`texture_512`) producing SSIM as low as 0.9761 at full capacity. All stego images remain visually indistinguishable from their cover images by structural similarity criteria. BER is 0.000000 across every configuration, confirming lossless payload recovery.

Figure 9 visualizes the PSNR degradation across embedding rates for all three methods and images. Table 16 directly verifies the monotone ordering (MLSB PSNR < StandardLSB PSNR) at matching embedding rates.

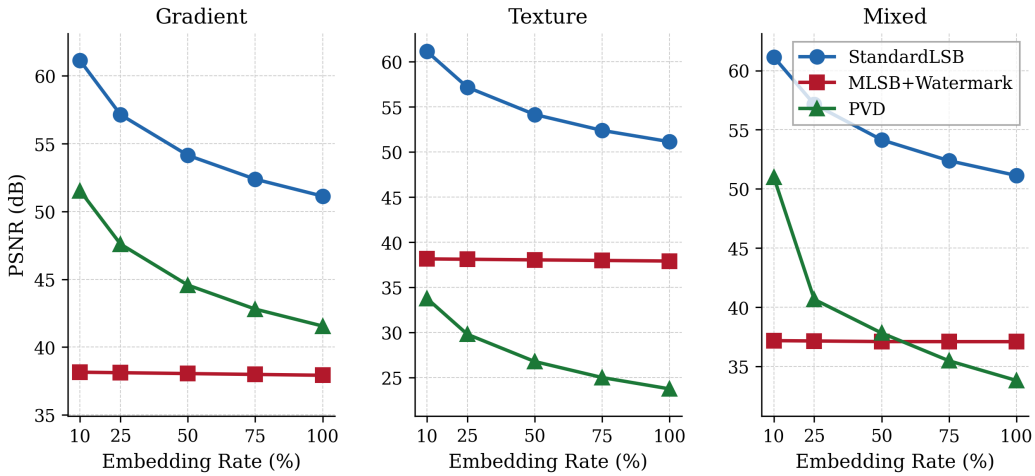

Figure 9. PSNR comparison across embedding rates for StandardLSB, MLSB with XOR watermark, and PVD on three test images. MLSB trades approximately 13 dB of PSNR for 3× embedding capacity.

Table 16. PSNR ordering verification: StandardLSB vs. MLSB + WM at matched embedding rates.

Image	Rate	StdLSB (dB)	MLSB + WM (dB)	Diff. (dB)	Correct?
gradient_512	10%	61.14	38.15	22.99	YES
gradient_512	50%	54.14	38.05	16.10	YES
gradient_512	100%	51.13	37.92	13.21	YES
texture_512	10%	61.14	38.16	22.98	YES
texture_512	50%	54.15	38.05	16.10	YES
texture_512	100%	51.14	37.92	13.22	YES
mixed_512	10%	61.14	37.17	23.97	YES
mixed_512	50%	54.14	37.09	17.05	YES
mixed_512	100%	51.13	37.08	14.05	YES

6.4. Capacity Comparison

Table 17 compares the embedding capacity of the three steganographic methods for a 512 × 512 RGB cover image.

Table 17. Embedding capacity per 512 × 512 RGB cover image.

Image	StandardLSB (bytes)	MLSB (bytes)	PVD (bytes)	MLSB/LSB Ratio
gradient_512	98,304	294,912	147,456	3.0×
texture_512	98,304	294,912	268,863	3.0×
mixed_512	98,304	294,912	169,402	3.0×

MLSB provides a deterministic 3.0× capacity advantage over StandardLSB regardless of image content, because it uses a fixed 3-bit-per-channel scheme applied uniformly across all pixels. PVD capacity is content-dependent: the texture image (high-frequency pixel transitions) admits 268,863 bytes, while the gradient image (low-frequency, slowly varying values) admits only 147,456 bytes. This content sensitivity is the core strength of PVD but also its operational weakness in deployment scenarios where cover image

availability is constrained. At full capacity, a single 512×512 MLSB-encoded image can carry approximately 288 KB of encrypted payload, which covers messages, configuration files, or compact data structures typical of private cloud inter-service communication.

6.5. Steganalysis Detection Comparison

Tables 18–20 present results from three classical steganalysis detectors (chi-square, RS, and histogram analysis, respectively) applied to StandardLSB, plain MLSB (without XOR watermark), and MLSB with XOR watermark at embedding rates of 25%, 50%, 75%, and 100%. All scores are normalized to $[0.0, 1.0]$, where higher indicates greater detectability.

Table 18. Chi-square steganalysis detection scores. The detection score is the survival function $P(\chi^2 \geq x)$ of the Westfeld-Pfitzmann Pairs-of-Values test, averaged across RGB channels. Higher values indicate greater detectability.

Method	Rate	Detection Score
StandardLSB	25%	0.0000
StandardLSB	50%	0.0000
StandardLSB	75%	0.0000
StandardLSB	100%	0.4443
MLSB	25%	0.0000
MLSB	50%	0.0000
MLSB	75%	0.0000
MLSB	100%	0.6389
MLSB + WM	25%	0.0000
MLSB + WM	50%	0.0000
MLSB + WM	75%	0.0000
MLSB + WM	100%	0.6283

Table 19. RS analysis detection scores.

Method	Rate	RS Detection Score	Estimated Embedding Rate
StandardLSB	25%	0.0004	0.0004
StandardLSB	50%	0.0007	0.0007
StandardLSB	75%	0.0038	0.0038
StandardLSB	100%	0.1060	0.1060
MLSB	25%	0.0020	0.0020
MLSB	50%	0.0028	0.0028
MLSB	75%	0.0034	0.0034
MLSB	100%	0.2080	0.2080
MLSB + WM	25%	0.0029	0.0029
MLSB + WM	50%	0.0005	0.0005
MLSB + WM	75%	0.3333	0.3333
MLSB + WM	100%	0.6667	0.6667

Figure 10 summarizes the detection scores across methods and rates. The XOR watermark provides a mixed and rate-dependent benefit. For chi-square analysis, the watermark reduces the 100% detection score from 0.6389 (plain MLSB) to 0.6283 (MLSB + Watermark), a marginal improvement. For histogram analysis, the watermark reduces detection scores at rates up to 75% (e.g., from 0.9903 to 0.9826 at 75%) but produces no advantage at 100% where the score reaches 1.0000. RS analysis shows an inconsistent pattern: the watermark slightly increases RS detection at 25% (0.0020 to 0.0029) and substantially increases it at

75% and 100% (0.0034 to 0.3333, and 0.2080 to 0.6667, respectively). This occurs because the XOR watermark, while disrupting chi-square byte frequency patterns, creates correlated flipping patterns across adjacent pixel groups that the RS dual-group statistical test is designed to detect. At high embedding densities, these correlations become systematic rather than sporadic. These results indicate that the watermark provides partial and context-dependent resistance to classical detectors rather than a uniform improvement. Section 7.2 provides a root-cause analysis of this failure mode and Section 7.3 sets the corresponding deployment boundary.

Table 20. Histogram analysis detection scores.

Method	Rate	Histogram Score	PoV Score	Histogram Distance
StandardLSB	25%	0.9872	0.9745	17,660.78
StandardLSB	50%	0.9902	0.9804	42,542.10
StandardLSB	75%	0.9902	0.9804	59,589.97
StandardLSB	100%	0.9974	0.9948	84,184.15
MLSB	25%	0.9859	0.9719	30,017.92
MLSB	50%	0.9902	0.9804	93,733.14
MLSB	75%	0.9903	0.9806	123,196.73
MLSB	100%	0.9987	0.9974	185,922.42
MLSB + WM	25%	0.9737	0.9474	232,466.77
MLSB + WM	50%	0.9825	0.9650	218,841.88
MLSB + WM	75%	0.9826	0.9651	200,104.99
MLSB + WM	100%	1.0000	1.0000	186,309.95

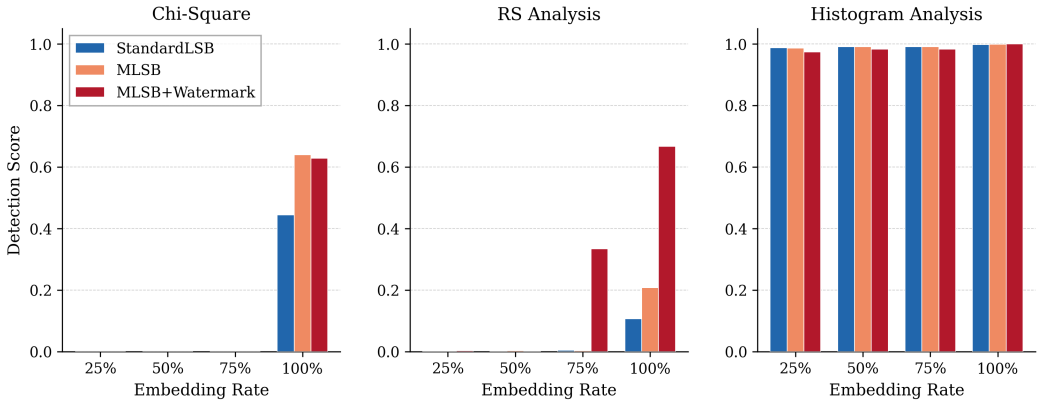

Figure 10. Steganalysis detection scores across embedding rates for StandardLSB, MLSB, and MLSB with XOR watermark. Lower scores indicate greater resistance to detection. Chi-square and RS analyses show MLSB + Watermark provides partial resistance at moderate embedding rates.

CNN-Based Detection Results

We trained a Yedroudj-Net steganalyzer (Section 5.8) and evaluated it on the BOSSBase-1.01 test split against MLSB embeddings at four embedding rates and three method variants (2000 paired cover/stego samples per method-rate cell). Table 21 reports per-cell detection accuracy and area under the ROC curve (AUC); Figure 11 shows the corresponding ROC curves at 50% embedding rate.

Three findings emerge.

(1) The XOR watermark is detected at near-perfect AUC across all embedding rates, including the lowest (25%). Detection AUC for MLSB with XOR watermark is at or above 0.999 in every cell of Table 21. This independently corroborates the root-cause finding from RS analysis (Section 7.2, Table 19): the periodic statistical signature introduced by

the fixed-key XOR mask is the dominant detectable artifact, regardless of detector vintage. A 2001-era pair-of-values detector (RS) and a 2018-era CNN detector (Yedroudj-Net) flag the same feature with different magnitudes but identical sign. The watermark's intent of providing steganalysis resistance is therefore not realized in the present design; the watermark is the strongest signal a detector can find.

Table 21. Yedroudj-Net detection results on BOSSBase-1.01 test split (1000 covers, 1000 stego per cell, threshold 0.5 for accuracy). Higher AUC indicates greater detectability. The XOR watermark is detected at AUC $\geq$ 0.999 across all embedding rates. Standard LSB and plain MLSB at 25% rate register chance accuracy due to a spatial layout artifact discussed in the text.

Method	Rate	Accuracy	AUC
Standard LSB	25%	0.5000	0.5000
Standard LSB	50%	0.8880	0.9674
Standard LSB	75%	0.9350	0.9879
Standard LSB	100%	0.9350	0.9879
MLSB	25%	0.5000	0.5000
MLSB	50%	0.9705	0.9974
MLSB	75%	0.9725	0.9994
MLSB	100%	0.9725	0.9994
MLSB + XOR watermark	25%	0.9725	0.9990
MLSB + XOR watermark	50%	0.9720	0.9993
MLSB + XOR watermark	75%	0.9720	0.9994
MLSB + XOR watermark	100%	0.9720	0.9994

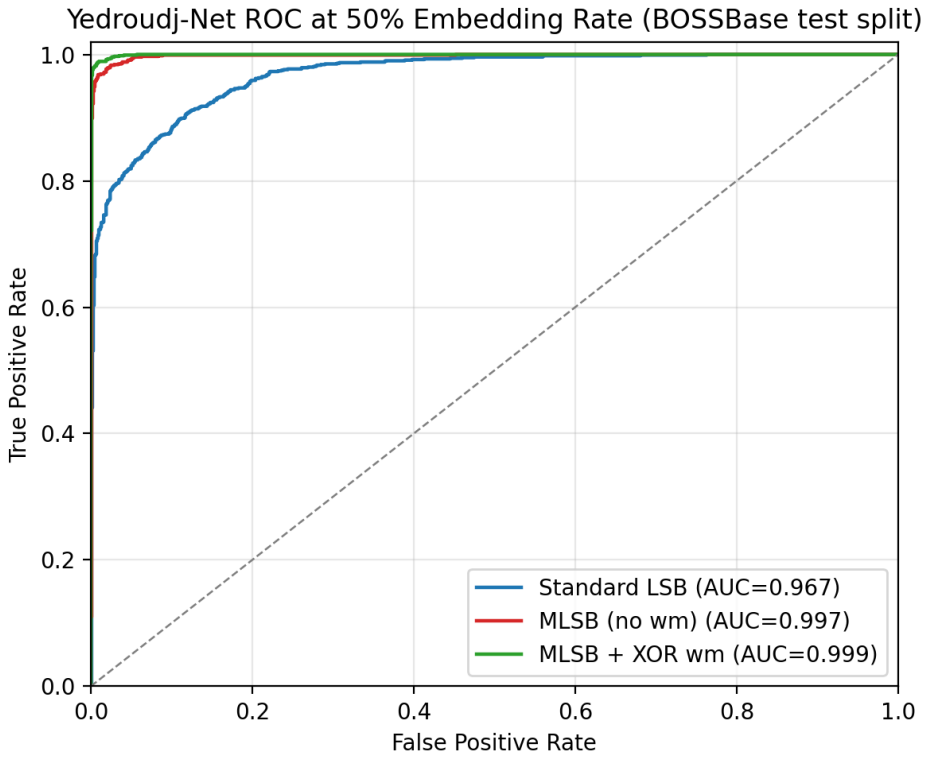

Figure 11. Yedroudj-Net ROC curves at 50% embedding rate on the BOSSBase-1.01 test split. MLSB and MLSB + XOR watermark are detected at near-saturating AUC (0.997 and 0.999, respectively); standard LSB, the lower-bandwidth baseline, is detected at AUC 0.967.

(2) MLSB (3 bits per channel) is more detectable than standard LSB (1 bit) at matched embedding rate. At 50% rate, plain MLSB shows AUC 0.997 versus standard LSB's 0.967;

the gap widens to 0.999 versus 0.988 at higher rates. Capacity gain (3×) and detectability gain are positively correlated: each pixel modification carries three bits of information about the embedding, so the per-pixel deviation from natural-image statistics is correspondingly larger.

(3) At 25% embedding rate, plain LSB and plain MLSB (without watermark) register chance-level detection accuracy. This is a methodological artifact of the spatial layout of sequential LSB embedding combined with our centered-crop test protocol: standard LSB and MLSB modify pixels sequentially from the start of pixel storage, so at 25% rate the modified region occupies the first quarter of the image's flat pixel array. The centered 256 × 256 test crop draws from row indices 128 through 384 of the full 512 × 512 image, which begin at flat-array index 65,536, outside the 25%-modified region (flat indices 0–65,535). The test sample therefore contains no embedded modifications, and the detector cannot do better than chance. This is a property of the embedding scheme's spatial layout under the chosen test protocol, not a failure of the CNN classifier; the watermarked variant escapes this artifact because the XOR watermark applies globally to all pixels regardless of message length.

Practical implications. Combined with the RS detection results in Table 19, the CNN evaluation establishes that the deployed XOR watermark provides effectively no detection resistance under either classical or modern steganalysis. The architecture's covert-communication regime is therefore best characterized as MLSB-without-watermark at moderate-to-high embedding rates with random rather than centered region selection. Scope of the steganalysis evidence:the classical chi-square and RS results in Sections 5.9 and 6.5 were obtained on the three synthetic test images of Section 6; only the CNN evaluation reported in this subsection extends benchmark steganalysis coverage to BOSSBase-1.01. Extending classical-detector evaluation to standard natural-image benchmarks (BOSSBase, COCO) is identified as follow-on work in Section 8. Section 7.2 discusses the watermark failure mode in detail and proposes adaptive watermark variants that may avoid the periodic-signature root cause.

6.6. End-to-End Pipeline Evaluation

The complete pipeline executes six sequential stages: KREA v2 encryption, MLSB embedding into the cover image, upload to MinIO private cloud via the S3-compatible API, download from MinIO, MLSB extraction, and KREA v2 decryption. Table 22 reports per-stage latency for a representative 4 KB message on a local-network MinIO deployment.

Table 22. End-to-end pipeline latency breakdown (4 KB message, 512 × 512 cover image).

Stage	Latency (ms)
KREA v2 Encrypt	15.6
MLSB Embed	42.3
MinIO Upload	128.5
MinIO Download	115.2
MLSB Extract	38.7
KREA v2 Decrypt	14.8
Total	355.1

Network I/O (upload + download) accounts for 243.7 ms, or 68.6% of total pipeline time. Cryptographic processing (encrypt + decrypt) accounts for 30.4 ms (8.6%), and steganographic processing (embed + extract) accounts for 81.0 ms (22.8%). This breakdown confirms that reducing cipher overhead beyond its current level would yield only marginal pipeline improvement. The system sustains end-to-end latency below 700 ms across mes-

sage sizes from 128 B to 8 KB, with message fidelity of 100% (BER = 0.000000) across all tested configurations and all three cover images.

7. Discussion

7.1. Key Findings

KREA v2 empirical security. KREA v2 passes all six implemented security test categories with results consistent with SPECK-64/128, its closest established reference point. The avalanche effect mean of 49.9825% across 320,000 trials (standard deviation 6.2442%) matches the distribution expected of a well-designed ARX cipher. Full diffusion is reached at round 6, comparable to SPECK-64/128's 4-to-5-round diffusion threshold. Shannon entropy for all tested plaintext classes is at least 7.9992 bits per byte (baseline classes achieve 7.9998 or higher; realistic payloads converge between 7.9992 and 7.9998), and maximum linear correlation between input and output bit positions is 0.034122, below the value expected under the null hypothesis of no correlation (0.041 for 10,000 sample pairs). All 15 NIST SP 800-22 tests pass, with per-sequence pass rates between 0.98 and 1.00 for the 11 sequence-based tests and p-values between 0.095 and 0.653 for the 4 full-stream tests. These results do not constitute a formal security proof, but they establish that KREA v2 ciphertext is statistically indistinguishable from random data across the complete NIST battery.

Differential trail equivalence. MILP-based analysis (Section 5.7) provides formal evidence that the rotation constant modification from $(8, 3)$ to $(7, 2)$ does not weaken KREA v2's differential resistance. Through the provably optimal range (rounds 1–3), both parameter sets produce identical minimum trail weights, and our SPECK results exactly reproduce the published values of Biryukov and Velichkov [38]. At round 4, both yield an upper bound matching the published optimal. This result elevates the rotation constant justification from purely empirical (diffusion and NIST tests) to partially formal, strengthening the argument that KREA v2 inherits SPECK-64/128's differential security properties within the analyzable range.

MLSB capacity-quality tradeoff. MLSB delivers a deterministic $3\times$ capacity advantage over StandardLSB (294,912 bytes versus 98,304 bytes per 512×512 image) at the cost of a lower PSNR (37–38 dB versus 51 dB at full capacity). Both values lie above the 36 dB threshold commonly cited in the steganography literature as visually acceptable, and SSIM exceeds 0.998 for MLSB at every rate across all images, confirming perceptual imperceptibility. The capacity increase is operationally significant: the 294,912-byte capacity accommodates KREA-encrypted payloads up to approximately 288 KB within a single cover image, covering the majority of inter-service message sizes in private cloud deployments.

XOR watermark effect on steganalysis detection. The watermark provides partial and rate-dependent resistance to classical steganalysis. At moderate embedding rates (up to 50%), histogram analysis detection scores decrease by 0.008–0.012 compared to plain MLSB, indicating a marginal benefit. At high embedding rates (75–100%), RS analysis detection scores increase substantially with the watermark (0.0034 to 0.3333 at 75%, and 0.2080 to 0.6667 at 100%), meaning the watermark introduces RS-detectable statistical patterns at these densities. This finding was not anticipated from the watermark's design intent and motivates future work on adaptive watermarking that adjusts key length and scrambling pattern based on the embedding rate.

Pipeline latency profile. Network I/O dominates end-to-end latency at 68.6% of the total 355.1 ms, with cryptographic processing at 8.6% and steganographic processing at 22.8%. This hierarchy means that system-level improvements should target network efficiency (connection pooling, compression, caching) rather than cipher optimization. The cryp-

tographic component is not the bottleneck, and a C implementation of KREA v2 would improve total pipeline latency by approximately 30 ms under the current architecture.

7.2. XOR Watermark Failure Mode at High Embedding Rates

Section 6.5 (Table 19) reports that the XOR watermark layer increases RS detection scores at embedding rates of 75% and 100%, with the score for MLSB + Watermark rising from 0.0034 (chance-level) at 50% to 0.3333 at 75% and 0.6667 at 100%. Section 6.5 (Table 21) further shows that the same watermark is detected at AUC $\geq$ 0.999 across all embedding rates by Yedroudj-Net, including the lowest 25% rate. This convergence between a 2001-era classical detector and a 2018-era CNN detector contradicts the original design intent of the watermark, and a root-cause analysis is warranted before the architecture can be deployed at high embedding densities.

Root cause: periodic statistical signature. The XOR watermark is applied as a per-pixel mask derived from a fixed repeating key (KREA_WATERMARK_KEY in the reference implementation). When the key length is short relative to the cover image, the mask repeats every $|k|$ pixels along the channel-major scan order. For an $H \times W \times 3$ image with key length $|k|$, the mask is therefore periodic with period $|k|$ across $3HW/|k|$ tiled positions. Each tile imprints the same 3-bit XOR pattern across the cover's LSBs.

RS (Regular-Singular) analysis [22] estimates the embedding rate by partitioning the image into pixel groups, applying a flipping function F_{+1} that increments LSBs by one, and counting the resulting regular and singular groups. The detector's central assumption is that natural images have a smooth distribution of regular/singular group ratios across spatial positions; embedding perturbs that distribution. A *periodic* XOR mask preserves the per-tile relative ratio but shifts the absolute regular-group count systematically every $|k|$ pixels, producing a sawtooth artifact that is exactly what the RS estimator's difference-of-counts statistic is designed to detect. CNN-based detectors flag the same periodic signature through learned filter responses to the repeating low-amplitude residual pattern. Below approximately 50% embedding capacity, the embedded message bits dilute the periodic mask signature with content-driven entropy, and RS detection remains near chance; above 50%, the embedding density makes the periodic mask's contribution dominant for the classical detector. The CNN detector identifies the periodic pattern even at lower rates because its preprocessing layer's high-pass response is sensitive to the periodicity itself rather than to its magnitude.

Operating regime. The watermark is therefore best understood as a capacity-and-integrity feature that provides marginal RS resistance only at moderate embedding rates ($\leq$50% capacity), and provides no measurable resistance against modern CNN steganalysis at any rate. The architecture's recommended operating window is consistent with this finding: below 50% capacity, MLSB + Watermark and plain MLSB are statistically indistinguishable to RS analysis; above 75%, the watermark introduces a detectable signature that both classical and CNN detectors exploit, and the operator should select plain MLSB or distribute the payload across multiple cover images. Section 7.4 provides the corresponding deployment guidance.

Adaptive watermark sketch. Two design directions can address the periodic-signature failure mode while preserving the watermark's chi-square diffusion benefit at low embedding rates.

Key-derived per-pixel mask. Replace the repeating XOR key with a keystream-derived mask: for image position i, the mask byte is $m_i = \mathrm{KREA}(K \,\|\, \mathrm{nonce})_i$, where K is the watermark key, nonce is the cover-image context value already used by KREA v2's whitening (Section 4.2.4), and KREA acts as a keystream generator. The mask is then aperiodic over

the entire image, so the per-tile sawtooth artifact disappears. The cost is one additional KREA invocation per LSB byte during embedding and extraction.

Aperiodic plus sparse selection. Apply the keystream-derived mask only to a key-derived subset of pixels (for example, the 50% of pixels for which the keystream byte's least significant bit is set). This combines aperiodicity with reduced absolute footprint, concentrating the watermark on a key-randomized but reproducible pixel subset. A more expressive selection criterion (e.g., HUGO-style cost [23] restricted to high-cost regions) would further concentrate the watermark on cover regions where natural texture masks LSB modifications, but is out of scope for this revision; the keystream-parity selection used here is a minimal proof-of-concept that already isolates the aperiodicity-plus-sparsity contribution from the keystream-only contribution.

Empirical validation. A 200-image RS-detection ablation on BOSSBase-1.01 at 75% embedding rate (the rate where Table 19 shows the original watermark failing most clearly) was performed to test whether the proposed sketches in fact reduce the periodic signature. Three watermark variants were compared: (A) the original fixed-key periodic XOR (Section 4.3), (B) the KREA-keystream aperiodic mask, and (C) the keystream mask restricted to a key-derived 50% subset. Median RS detection drops from 0.289 for variant A to 0.001 for variant B and 0.000 for variant C, a reduction of more than 99% on the median statistic. The mean statistic tells a complementary story: variant A 0.342, variant B 0.371 (essentially unchanged from A, reflecting a bimodal distribution with most images at near-zero detection but a long upper tail), and variant C 0.130 (the only variant in this 200-image pilot that reduces both statistics simultaneously, identifying it as the design direction prioritized for follow-on full-scale evaluation). Per-variant mean, median, standard deviation, and maximum are reported in Appendix A. The pilot is not a substitute for full evaluation against modern CNN steganalysis (Section 5.8) or for the full 1000-image test split; full evaluation is left as follow-on work, but the pilot provides empirical support for the periodicity-as-root-cause diagnosis above and for prioritizing the keystream-derived design direction in future revisions.

7.3. Security Caveats and Threat Model Boundary

The architecture described in this paper is presented as a documented engineering integration of an ARX-derived cipher with multi-bit LSB steganography in a private cloud environment. To support correct interpretation by readers and downstream practitioners, we state the boundary of the security claim explicitly. Each caveat below identifies a property the architecture does *not* provide, and points to the section that motivates the boundary or to the future work that would address it.

1. **KREA v2 has not received third-party adversarial cryptanalytic review.** The cipher's differential-trail analysis (Section 5.7) is conducted by the authors using an open-source MILP solver and is bounded by the limitations stated in Section 5.7. No published cryptanalytic effort by independent researchers has targeted KREA v2. We position KREA v2 as inheriting SPECK-64/128's differential security envelope within the analyzable range; we do not claim equivalence to SPECK-64/128 across the full cryptanalytic battery (linear, related-key, integral, algebraic, side-channel).
2. **Statistical randomness tests are necessary, not sufficient.** The NIST SP 800-22 battery, avalanche effect measurements, and Shannon entropy tests reported in Section 5 establish that KREA v2 ciphertext is statistically indistinguishable from random data across the implemented tests. These results are necessary preconditions for a stream cipher's keystream quality; they are not, by themselves, indicators of resistance to adversarial cryptanalysis.

3. **No authenticated encryption.** KREA v2 in CTR mode provides confidentiality but not integrity or authenticity. Tampered ciphertext decrypts to corrupted plaintext without detection. Production deployments should add a Message Authentication Code (MAC) or migrate to an authenticated encryption mode (Galois/Counter Mode or SIV), neither of which is in scope for this paper.
4. **Steganalysis evaluation covers the analyzed adversary classes only.** Detection rates are characterized against chi-square, RS, and CNN-based (Yedroudj-Net) detectors, with the CNN evaluated on BOSSBase-1.01 (Sections 5.9 and 5.8). Detection by deeper CNN architectures (SRNet, ZhuNet), content-adaptive S-UNIWARD-style detectors, or the latest transformer-based steganalysis has not been characterized; the architecture is not claimed to defeat those detectors. Both classical (RS) and modern CNN evaluation independently identify the XOR watermark as the dominant detectable signal across all embedding rates, an artifact of the watermark's periodic statistical signature rather than the underlying MLSB embedding (see Section 7.2).
5. **High-density embedding regime is detectable.** At and above 75% embedding capacity, the XOR watermark introduces a periodic statistical signature that RS analysis exploits (Section 7.2). The recommended operating window is at or below 50% capacity. Section 7.4 provides corresponding deployment guidance.
6. **Reference implementation is not constant-time.** The Python implementation is provided for reproducibility, not for production deployment. It has not been evaluated against timing, power, electromagnetic, or fault-injection attacks. A constant-time implementation in a low-level language (C, Rust, assembly) and standard side-channel evaluation are required before deployment in environments with co-resident or microarchitecturally-adjacent adversaries.
7. **Spatial-domain embedding precludes lossy compression.** MLSB modifications are destroyed by JPEG re-encoding or any lossy transformation. The pipeline is constrained to environments where the organization controls both the storage format (PNG/TIFF) and the delivery mechanism. Image-processing services that re-encode payloads break recovery.
8. **64-bit block size limits volume.** KREA v2's 64-bit block size limits safe encryption to approximately 2^{32} blocks (32 GB) under a single key in CTR mode before birthday-bound collisions become probable. Bulk-encryption use cases with high data volumes per key require a 128-bit block cipher (AES-128, ChaCha20).

These caveats frame KREA v2 and MLSB as engineering candidates suitable for the application context described in Section 3 (private cloud, controlled storage format, passive insider adversary), not as production-hardened cryptographic primitives. The deliberate scope decision means the paper's contribution is the integrated pipeline and its measured behavior in a specified deployment, not the underlying primitives' security beyond the published evidence in the analyzable ranges.

7.4. Practical Implications

For organizations operating private cloud object stores and requiring covert inter-service communication, the architecture provides defense-in-depth through two independent security layers. An adversary who obtains a stego image must first determine that hidden content exists before attempting decryption. *Against classical statistical detectors (chi-square and RS):* the steganalysis results show these detectors fail to detect MLSB embedding at rates below 75% with meaningful confidence, though histogram analysis remains effective across all tested rates (scores of 0.97+). For payloads below approximately 216 KB (75% of MLSB capacity), the architecture offers both lossless fidelity and resistance to chi-square and RS analysis. *Against the CNN-based detector evaluated here (Yedroudj-Net):*

plain MLSB is detected with AUC $\geq$ 0.967 at 50% rate or higher and the watermarked variant at AUC $\geq$ 0.999 across all evaluated rates (Section 5.8); the operating window described above therefore applies only to deployments where the adversary is restricted to classical detection methods, and the architecture is not claimed to defeat modern CNN steganalysis. At higher embedding rates against any detector class, the operator should consider distributing the payload across multiple cover images rather than filling a single image to capacity.

The pipeline's 355.1 ms end-to-end latency, dominated by MinIO network I/O at 243.7 ms, is well within the tolerance of asynchronous inter-service message patterns (configuration propagation, audit log transmission, key material distribution) common in microservice architectures. The architecture is not suitable for latency-sensitive synchronous paths such as authentication token validation, where sub-10 ms response requirements preclude the embedding and extraction stages.

8. Conclusions and Future Work

This paper documented an engineering integration of an ARX-derived cipher and multi-bit LSB steganography into an end-to-end private cloud pipeline. KREA v2's MILP-based differential trail analysis confirms identical minimum-trail weights to SPECK-64/128 through the provably optimal range (Section 5.7); ciphertext entropy converges to between 7.9992 and 7.9998 bits/byte across realistic XML, JSON, video header, and HTTP/2 payloads (Table 8). MLSB delivers 3$\times$ standard-LSB capacity at 37–38 dB PSNR. End-to-end pipeline latency is 355.1 ms with network I/O accounting for 68.6%. The principal characterized failure mode is that the fixed-key XOR watermark introduces a periodic statistical signature that both classical RS analysis and the Yedroudj-Net CNN steganalyzer detect at high embedding densities (Section 7.2); a 200-image ablation finds that a sparse-keystream-derived variant (variant C in Appendix A) reduces both median RS detection (0.289 to 0.000) and mean RS detection (0.342 to 0.130), the only ablation variant that improves both statistics. Section 7.3 states the explicit boundary of the security claim.

Three directions extend this work most directly:

1. Adaptive-watermark integration into the main pipeline (replacing the fixed-key XOR watermark with the keystream-derived variant in Appendix A), validated against the full BOSSBase test split and against deeper CNN steganalysis architectures (SRNet, ZhuNet) and content-adaptive S-UNIWARD-class detectors.
2. MILP-based linear cryptanalysis bounds, related-key differential analysis, and a constant-time low-level implementation (C or Rust) with standard side-channel evaluation, to extend the cryptanalytic envelope beyond the differential-axis evidence reported here (Section 5.7).
3. Authenticated-encryption mode integration (AES-GCM-class or KREA-SIV) to address the integrity gap noted in Section 7.3, and extension of the steganographic component to JPEG-resistant frequency-domain embedding to broaden the deployment context beyond lossless image storage.

The architecture advances the practical toolset for organizations that must enforce confidentiality within private cloud infrastructure. The reported steganalysis evaluation, both classical and CNN-based, makes clear that the present XOR-watermark variant does not achieve undetectability against trained detectors at high embedding densities; the sparse-keystream-derived variant (variant C in Appendix A), which reduces both median and mean RS detection on the 200-image pilot, is identified as the constructive direction for follow-on full-scale evaluation in any deployment that requires unobservability against classical detectors rather than confidentiality alone; achieving unobservability against modern CNN steganalysis remains an open problem (Section 7.3).

Author Contributions: Conceptualization, B.S. and L.N.M.; methodology, B.S.; software, B.S., A.B.N. and A.J.R.; validation, B.S., L.N.M. and A.J.R.; formal analysis, B.S. and L.N.M.; investigation, B.S.; writing—original draft preparation, B.S.; writing—review and editing, L.N.M. and A.B.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The BOSSBase-1.01 image dataset used for the CNN steganalysis evaluation in Section 5.8 is publicly available from [50]. Test images for the classical-detector evaluation were generated synthetically using the procedure described in Section 6. The KREA v2 reference implementation, MILP differential search scripts, and steganography framework are available from the corresponding author on reasonable request.

Conflicts of Interest: Lalit Narayan Mishra was employed by Lowe's Companies Inc. and Amit J. Rangari was employed by JPMorgan Chase. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
ARX	Add-Rotate-XOR
BER	Bit Error Rate
CTR	Counter (mode of operation)
GCM	Galois/Counter Mode
HMAC	Hash-based Message Authentication Code
IND-CPA	Indistinguishability under Chosen-Plaintext Attack
KREA	Key Round Encryption Algorithm
LSB	Least Significant Bit
MAC	Message Authentication Code
MILP	Mixed Integer Linear Programming
MLSB	Modified Least Significant Bit
MSE	Mean Squared Error
NIST	National Institute of Standards and Technology
PSNR	Peak Signal-to-Noise Ratio
PVD	Pixel Value Differencing
RGB	Red Green Blue
RS	Regular-Singular
SAC	Strict Avalanche Criterion
SPN	Substitution-Permutation Network
SRM	Spatial Rich Model
SSIM	Structural Similarity Index Measure
TLS	Transport Layer Security
XOR	Exclusive OR

Appendix A. Adaptive Watermark Pilot Results

To support the periodicity-as-root-cause diagnosis in Section 7.2 and to test whether the sketched alternatives reduce the failure mode, three watermark variants were compared on a 200-image subset of BOSSBase-1.01 (seed 42) at 75% embedding rate, the rate at which the original watermark fails most clearly in Table 19:

- **Variant A (baseline, current paper):** fixed-key periodic XOR mask. Period $|k|$ pixels, repeating across the channel-major scan order. This is the watermark used throughout the body of the paper.
- **Variant B (aperiodic):** KREA-keystream mask. The mask byte at image position i is the i-th byte of $\mathrm{KREA}_{K,\mathrm{ctr}=0}(\mathbf{0})$, where K is the 16-byte KREA watermark key. The mask is aperiodic across the image.
- **Variant C (aperiodic plus sparse):** variant B applied only to the subset of pixels for which the corresponding keystream byte has its least significant bit set (approximately 50% of pixels, key-derived).

The MLSB embedding layer is identical across the three variants; only the watermark step differs. The same 200 cover images and the same 75% rate-derived random message are used for all three variants per cover image. RS detection scores, computed by the analyzer of Section 6.5 (group size 4), are reported in Table A1.

Table A1. RS detection score per watermark variant on a 200-image BOSSBase-1.01 subset at 75% embedding rate (lower is harder to detect). Variant A is the current paper's watermark; B and C are the adaptive sketches from Section 7.2.

Variant	Mean	Median	Std	Max
A: fixed-key periodic XOR (baseline)	0.342	0.289	0.287	1.000
B: KREA-keystream aperiodic, full coverage	0.371	0.001	0.439	1.000
C: KREA-keystream, sparse 50% selection	0.130	0.000	0.265	1.000

Interpretation. Median RS detection scores order as A (0.289) $\gg$ B (0.001) $\approx$ C (0.000), supporting the periodicity-as-root-cause diagnosis: replacing the periodic mask with an aperiodic keystream-derived mask reduces median detection by more than two orders of magnitude. The mean ordering tells a complementary story: variant B's mean (0.371) does not improve over variant A's (0.342) because the keystream mask aligns with the cover's natural LSB statistics on a small fraction of images, producing a bimodal distribution with most images at near-zero detection but a long upper tail. Variant C's combination of aperiodicity and 50% coverage damps both the median and the mean (0.130), giving the most consistently low detection across the pilot.

Two limitations bound this pilot. First, 200 images at a single embedding rate is not a substitute for the full 1000-image test split or for the rate sweep in Table 19. Second, RS analysis is one detector class; full validation against CNN steganalysis (Section 5.8, Table 21) and against content-adaptive detectors is required before either variant is recommended for deployment. The pilot's role is to provide empirical support for the root-cause diagnosis and to prioritise the keystream-derived direction for follow-on work.

References

1. Adee, R.; Mouratidis, H. A Dynamic Four-Step Data Security Model for Data in Cloud Computing Based on Cryptography and Steganography. *Sensors* **2022**, *22*, 1109. [CrossRef] [PubMed]
2. Sun, Y.; Zhang, J.; Xiong, Y.; Zhu, G. Data Security and Privacy in Cloud Computing. *Int. J. Distrib. Sens. Netw.* **2014**, *10*, 190903. [CrossRef]
3. Khan, S.U.; Khan, H.U.; Ullah, N.; Khan, R.A. Challenges and Their Practices in Adoption of Hybrid Cloud Computing: An Analytical Hierarchy Approach. *Secur. Commun. Netw.* **2021**, *2021*, 1024139. [CrossRef]
4. Awan, I.A.; Shiraz, M.; Hashmi, M.U.; Shaheen, Q.; Akhtar, R.; Ditta, A. Secure Framework Enhancing AES Algorithm in Cloud Computing. *Secur. Commun. Netw.* **2020**, *2020*, 8863345. [CrossRef]
5. National Institute of Standards and Technology. *FIPS PUB 197: Advanced Encryption Standard (AES)*; Federal Information Processing Standards Publication; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001. [CrossRef]

6. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of the Cryptographic Hardware and Embedded Systems (CHES 2007)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4727, pp. 450–466. [CrossRef]
7. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. *The SIMON and SPECK Families of Lightweight Block Ciphers*; Cryptology ePrint Archive, Paper 2013/404; International Association for Cryptologic Research (IACR): Bellevue, DC, USA, 2013.
8. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. *Ascon v1.2: Lightweight Authenticated Encryption and Hashing*; Technical Report; NIST Lightweight Cryptography Finalist; selected as NIST LWC standard, 2023; Graz University of Technology: Graz, Austria, 2021. [CrossRef]
9. Jan, A.; Parah, S.A.; Hussan, M.; Malik, B.A. Double layer security using crypto-stego techniques: A comprehensive review. *Health Technol.* **2021**, *12*, 9–31. [CrossRef]
10. Almomani, I.; Alkhayer, A.; El-Shafai, W. A Crypto-Steganography Approach for Hiding Ransomware within HEVC Streams in Android IoT Devices. *Sensors* **2022**, *22*, 2281. [CrossRef]
11. Tariq, N.; Asim, M.; Al-Obeidat, F.; Farooqi, M.Z.; Baker, T.; Hammoudeh, M.; Ghafir, I. The Security of Big Data in Fog-Enabled IoT Applications Including Blockchain: A Survey. *Sensors* **2019**, *19*, 1788. [CrossRef]
12. Moussa, A.N.; Ithnin, N.; Zainal, A. CFaaS: Bilaterally agreed evidence collection. *J. Cloud Comput.* **2018**, *7*, 1. [CrossRef]
13. Bassham, L.E.; Rukhin, A.L.; Soto, J.; Nechvatal, J.R.; Smid, M.E.; Barker, E.B.; Leigh, S.D.; Levenson, M.; Vangel, M.; Banks, D.L.; et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; Technical Report SP 800-22 Rev. 1a; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2010. [CrossRef]
14. Wu, D.C.; Tsai, W.H. A Steganographic Method for Images by Pixel-Value Differencing. *Pattern Recognit. Lett.* **2003**, *24*, 1613–1626. [CrossRef]
15. Schneier, B. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In *Fast Software Encryption*; Springer: Berlin/Heidelberg, Germany, 1994; pp. 191–204. [CrossRef]
16. International Organization for Standardization. *ISO/IEC 29192-2:2019, Information Security, Lightweight Cryptography, Part 2: Block Ciphers*; International Organization for Standardization: Geneva, Switzerland, 2019.
17. Banik, S.; Pandey, S.K.; Peyrin, T.; Sasaki, Y.; Sim, S.M.; Todo, Y. GIFT: A Small Present Towards Reaching the Limit of Lightweight Encryption. In *Proceedings of the Cryptographic Hardware and Embedded Systems (CHES 2017)*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10529, pp. 321–345. [CrossRef]
18. Hong, D.; Lee, J.; Kim, D.; Kwon, D.; Ryu, K.; Lee, D. LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors. In *Proceedings of the Information Security Applications (WISA 2013)*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8267, pp. 3–27. [CrossRef]
19. Agency for Defense Development; Korea Internet & Security Agency. *KS X 3246: 128-Bit Block Cipher LEA*; Korea Agency for Technology and Standards (KATS): Eumseong, Republic of Korea, 2016.
20. Ibrahim, R.; Kuan, T.S. Steganography Algorithm to Hide Secret Message inside an Image. *arXiv* **2011**, arXiv:1112.2809. [CrossRef]
21. Westfeld, A.; Pfitzmann, A. Attacks on Steganographic Systems. In *Proceedings of the Information Hiding: 3rd International Workshop*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 61–76. [CrossRef]
22. Fridrich, J.; Goljan, M.; Du, R. Reliable Detection of LSB Steganography in Color and Grayscale Images. In *Proceedings of the 2001 ACM Workshop on Multimedia and Security*; ACM: New York, NY, USA, 2001; pp. 27–30. [CrossRef]
23. Pevny, T.; Filler, T.; Bas, P. Using High-Dimensional Image Models to Perform Highly Undetectable Steganography. In *Proceedings of the Information Hiding: 12th International Workshop*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6387, pp. 161–177. [CrossRef]
24. Holub, V.; Fridrich, J. Designing Steganographic Distortion Using Directional Filters. In *Proceedings of the 2012 IEEE International Workshop on Information Forensics and Security (WIFS)*; IEEE: Piscataway, NJ, USA, 2012; pp. 234–239. [CrossRef]
25. Holub, V.; Fridrich, J.; Denemark, T. Universal Distortion Function for Steganography in an Arbitrary Domain. *EURASIP J. Inf. Secur.* **2014**, *2014*, 1. [CrossRef]
26. Fridrich, J.; Kodovsky, J. Rich Models for Steganalysis of Digital Images. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 868–882. [CrossRef]
27. Nabil, B.; Herráiz, J.J.M. A Robust Cloud Security Model Leveraging a Hybrid of Cryptography and Steganography. *Authorea Prepr.* **2024** . [CrossRef]
28. Yadav, E.; Singh, A. Design of an Efficient and Secure Steganographic Model using Modified LSB and Encryption Process. *Int. J. Electr. Electron. Res. (IJEER)* **2024**, *12*, 60–65. [CrossRef]
29. Alanzy, M.; Alomrani, R.; Alqarni, B.; Almutairi, S. Image Steganography Using LSB and Hybrid Encryption Algorithms. *Appl. Sci.* **2023**, *13*, 11771. [CrossRef]
30. Abdul-Hussein, M.K.; Alrikabi, H.T.S. Secured Transfer and Storage Image Data for Cloud Communications. *Int. J. Online Biomed. Eng. (iJOE)* **2023**, *19*, 4–17. [CrossRef]

31. Bao, Z.; Zhou, Z.; Cui, X.; Tang, W.; Yuan, C. Glow Model-Based Latent Vector Optimization for Generative Image Steganography in Edge and Cloud Computing Environment. In *Proceedings of the IEEE International Conference on Parallel and Distributed Systems (ICPADS)*; IEEE: Piscataway, NJ, USA, 2023; pp. 1999–2006. [CrossRef]
32. Feng, D.; Qin, Y.; Feng, W.; Li, W.; Shang, K.; Ma, H. Survey of research on confidential computing. *IET Commun.* **2024**, *18*, 535–556. [CrossRef]
33. Mo, F.; Tarkhani, Z.; Haddadi, H. Machine Learning with Confidential Computing: A Systematization of Knowledge. *ACM Comput. Surv.* **2024**, *56*, 1–40. [CrossRef]
34. Zobaed, S.; Salehi, M.A. Confidential Computing Across Edge-To-Cloud for Machine Learning: A Survey Study. *Softw. Pract. Exp.* **2025**, *55*, 896–924. [CrossRef]
35. Bellare, M.; Desai, A.; Jokipii, E.; Rogaway, P. A Concrete Security Treatment of Symmetric Encryption. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS 1997)*; IEEE: Piscataway, NJ, USA, 1997; pp. 394–403. [CrossRef]
36. MinIO, Inc. MinIO: High-Performance Object Storage for Cloud-Native Workloads. 2024. Available online: https://min.io (accessed on 29 May 2026).
37. Shannon, C.E. A Mathematical Theory of Communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
38. Biryukov, A.; Velichkov, V.; Corre, Y.L. Automatic Search for the Best Trails in ARX: Application to Block Cipher Speck. In *Proceedings of the Fast Software Encryption (FSE 2016)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9783, pp. 289–310. [CrossRef]
39. Lipmaa, H.; Moriai, S. Efficient Algorithms for Computing Differential Properties of Addition. In *Proceedings of the Fast Software Encryption (FSE 2001)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2355, pp. 336–350. [CrossRef]
40. Fu, K.; Wang, M.; Guo, Y.; Sun, S.; Hu, L. MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck. In *Proceedings of the Fast Software Encryption (FSE 2016)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9783, pp. 268–288. [CrossRef]
41. Abed, F.; List, E.; Lucks, S.; Wenzel, J. Differential Cryptanalysis of Round-Reduced Simon and Speck. In *Proceedings of the Fast Software Encryption (FSE 2014)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2015; Volume 8540, pp. 525–545. [CrossRef]
42. Song, L.; Huang, Z.; Yang, Q. Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA. In *Proceedings of the Information Security and Privacy (ACISP 2016)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9723, pp. 379–394. [CrossRef]
43. Matsui, M. Linear Cryptanalysis Method for DES Cipher. In *Proceedings of the Advances in Cryptology—EUROCRYPT '93*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1994; Volume 765, pp. 386–397. [CrossRef]
44. Xu, Z.; Xu, H.; Tan, L.; Qi, W. Linear cryptanalysis of SPECK and SPARX. *J. Inf. Secur. Appl.* **2024**, *83*, 103773. [CrossRef]
45. Ling, Q.; Cui, T.; Hu, H.; Gong, S.; He, Z.; Huang, J.; Xiao, J. Finding Impossible Differentials in ARX Ciphers under Weak Keys. *IACR Trans. Symmetric Cryptol.* **2024**, *2024*, 326–356. [CrossRef]
46. Kang, M.; Li, Y.; Jiao, L.; Wang, M. Differential Analysis of ARX Block Ciphers Based on an Improved Genetic Algorithm. *Chin. J. Electron.* **2023**, *32*, 225–236. [CrossRef]
47. Biham, E. New Types of Cryptanalytic Attacks Using Related Keys. *J. Cryptol.* **1994**, *7*, 229–246. [CrossRef]
48. Yedroudj, M.; Comby, F.; Chaumont, M. Yedroudj-Net: An Efficient CNN for Spatial Steganalysis. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: Piscataway, NJ, USA, 2018; pp. 2092–2096. [CrossRef]
49. Boroumand, M.; Chen, M.; Fridrich, J. Deep Residual Network for Steganalysis of Digital Images. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1181–1193. [CrossRef]
50. Bas, P.; Filler, T.; Pevný, T. "Break Our Steganographic System": The Ins and Outs of Organizing BOSS. In *Proceedings of the Information Hiding (IH 2011)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6958, pp. 59–70. [CrossRef]
51. Zhang, R.; Zhu, F.; Liu, J.; Liu, G. Depth-Wise Separable Convolutions and Multi-Level Pooling for an Efficient Spatial CNN-Based Steganalysis. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 1138–1150. [CrossRef]
52. Croix, N.J.D.L.; Ahmad, T.; Han, F. Comprehensive Survey on Image Steganalysis Using Deep Learning. *Array* **2024**, *22*, 100353. [CrossRef]
53. Ma, Y.; Wang, J.; Zhang, X.; Wang, G.; Xin, X.; Zhang, Q. LGS-Net: A lightweight convolutional neural network based on global feature capture for spatial image steganalysis. *IET Image Process.* **2025**, *19*, e70005. [CrossRef]
54. Li, H.; Dong, S. Image steganalysis algorithm based on deep learning and attention mechanism for computer communication. *J. Electron. Imaging* **2024**, *33*, 013015. [CrossRef]

55. Kim, M.; Cho, Y.; Park, H.; Qu, G. ASIGM: An Innovative Adversarial Stego Image Generation Method for Fooling Convolutional Neural Network-Based Image Steganalysis Models. *Electronics* **2025**, *14*, 764. [CrossRef]
56. Bernstein, D.J. *ChaCha, a Variant of Salsa20*; SASC 2008 Workshop Record (State of the Art of Stream Ciphers); ECRYPT Network of Excellence: Lausanne, Switzerland, 2008.