International Journal of Advanced Electrical and Electronics Engineering

ISSN: 2278-8948

Volume 15 Issue 2 (2026) | Pages: 9–23

Text-To-Visual Cryptographic Sharing: A Lightweight Framework For Secure Agent Communication

Piyush Indorkar[1], Amarsinh Varpe[2]

[12]School of Computer Science and Engineering, Sandip University, Nashik, Maharashtra, India

[1]piyush121195@gmail.com, [2] amarsinh.varpe@sandipuniversity.edu.in

Peer Review Information

***Type**: Article*
***Received:** 15Aug 2026*
***Revised:** 22 Sep 2026*
***Accepted:** 20 Oct 2026*
***Published:** 18 Nov 2026*

Abstract

Secure communication in constrained and intermittently connected environments requires more than confidentiality: message size, key distribution, device energy, channel availability, integrity, and the detectability of communication also influence operational suitability. Visual cryptography (VC) offers a threshold secret-sharing approach in which a secret visual pattern is distributed among shares and reconstructed when an authorized subset is combined. This paper examines a text-to-visual-share (T2VS) workflow derived from a dissertation prototype that converts short ASCII messages into binary sequences, expands bits into subpixel patterns, distributes two shares, and reconstructs the message through a deterministic combining operation. The source dissertation reports a synthetic corpus of 500 messages, exact reconstruction for all test cases, and mean encoding times of 0.11 ms, 0.36 ms, and 0.72 ms for 30-, 100-, and 200-character messages, respectively. These figures are presented as source-reported prototype results, not as independently replicated measurements. A critical review identifies a fundamental security issue in the deterministic encoding rule used by the prototype: encoding a zero bit as [0,0] and a one bit as either [1,0] or [0,1] makes the distribution of an individual share depend on the secret bit. Consequently, the stated rule does not provide perfect secrecy for a single share, and its share patterns are statistically distinguishable. The paper therefore separates the reproduced baseline from a corrected research direction based on standard visual-secret-sharing basis matrices, in which each participant's marginal share distribution is independent of the secret and reconstruction is based on contrast/threshold decoding rather than a simplistic OR-to-bit mapping. The manuscript develops the system model, threat model, encoding and decoding workflow, evaluation methodology, limitations, and future research agenda. It argues that T2VS should be treated as a specialized secret-sharing layer—not a replacement for authenticated encryption—and recommends combining it with integrity protection, error correction, careful channel separation, and reproducible security testing.

Keywords: visual cryptography; visual secret sharing; text-to-visual-share; threshold secret sharing; secure communication; ASCII encoding; information security; share generation; steganalysis.

How to Cite This Article

Indorkar, P. Varpe, A. (2026). Text-To-Visual Cryptographic Sharing: A Lightweight Framework for Secure Agent Communication. *International Journal of Advanced Electrical and Electronics Engineering 15*(2), 9–23

2278-8948/ © 2015 The Authors. This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0).

Introduction

Digital communication is now embedded in field operations, emergency response, industrial control, remote sensing, and other settings in which short messages may be transmitted over networks that cannot be assumed trustworthy. Messages can contain coordinates, status indicators, short instructions, or alerts. Conventional cryptography protects message content when correctly implemented and key material is protected, but encrypted traffic can still reveal communication metadata such as timing, endpoints, and volume. In addition, cryptographic systems require key establishment, key rotation, storage, and recovery procedures. These are not reasons to avoid modern cryptography; they are engineering constraints that motivate complementary techniques.

Visual cryptography was introduced by Naor and Shamir as a visual form of threshold secret sharing. A secret image is encoded into multiple shares, and a qualified subset reveals the secret when the shares are superimposed. The original construction was designed so that unqualified subsets obtain no information about the secret, under the formal model of the scheme [1]. Subsequent work extended the framework to general access structures, grayscale images, meaningful shares, halftone images, and schemes with improved contrast or reduced pixel expansion [3]–[12].

The dissertation supplied for this work proposes a text-oriented adaptation called Text-to-Visual-Share (T2VS). It maps each character to an 8-bit ASCII representation and creates two binary arrays that are rendered as image shares. The intended application is short-message exchange where the two shares can be delivered over distinct channels and recombined by the receiver. The direct mapping from text bits to share symbols is an attractive implementation idea because it avoids rasterizing a complete text image before applying visual cryptography.

However, adapting a visual cryptography construction to text does not automatically preserve the security properties of the original construction. A valid secret-sharing scheme must ensure that the distribution of every forbidden set of shares is independent of the secret. The prototype's deterministic zero-bit mapping violates this requirement. This paper makes that issue explicit and uses it to define a more defensible framework for further work.

1.1 Problem Statement

The research problem is to design and evaluate a compact, text-oriented share-generation pipeline that supports exact reconstruction under ideal conditions while avoiding leakage from any single share. The design must also account for transmission errors, share misalignment, integrity, message-length leakage, channel compromise, and the operational cost of distributing two separate objects.

1.2 Aim and Objectives

- Translate short textual messages into a reversible binary representation suitable for visual-secret-sharing operations.
- Generate two shares using a formally specified (2,2) visual-secret-sharing construction rather than an ad hoc bit expansion.
- Define receiver-side reconstruction, alignment, validation, and error handling.
- Analyze confidentiality, share leakage, integrity, availability, traffic analysis, and steganalysis separately.
- Compare the framework with authenticated encryption and conventional steganography without claiming that the approaches provide identical security services.
- Establish a reproducible experimental plan for latency, storage overhead, reconstruction error, randomness, and adversarial evaluation.

1.3 Contributions and Scope

The paper contributes (i) a structured synthesis of the T2VS prototype described in the supplied dissertation; (ii) a technical audit of its stated share-generation rule; (iii) a corrected basis-matrix design direction; (iv) a system and threat model; and (v) an evaluation protocol that distinguishes source-reported results from results that still require independent replication. No new experiment is claimed in this manuscript.

Background and Theoretical Foundations

2.1 Visual Cryptography and Threshold Secret Sharing

A (k,n)-threshold visual cryptography scheme distributes a secret among n participants so that any qualified set of at least k shares can reconstruct the secret, while any forbidden set of fewer than k shares reveals no information in the information-theoretic model. In a basic binary scheme, each secret pixel is expanded into m subpixels in every share. The reconstructed darkness depends on the Hamming weight of the superimposed subpixel pattern. Pixel expansion, contrast, and the distributions of share patterns are central design parameters [1], [5], [9].

The important distinction is between a share that merely looks noisy and a share whose probability distribution is independent of the secret. Random appearance is a visual impression; secrecy is a mathematical property. A pattern can look irregular to a person and still leak information through local symbol frequencies, run lengths, or other statistics.

2.2 Text-to-Binary Representation

The source prototype uses printable ASCII characters (codes 32–126), converting each character to eight bits. For a message M containing L ASCII characters, the bitstream length is B = 8L. ASCII is adequate for an English-only prototype but is not a general

Unicode encoding. A production system should explicitly choose UTF-8, specify normalization rules, and encode the byte length and version in a protected header [24].

For example, the ASCII character 'A' has decimal value 65 and binary representation 01000001. The byte stream is then supplied to the share encoder. The receiver reverses the process only after validating share dimensions, message length, and integrity metadata.

2.3 Secret Sharing Versus Encryption and Steganography

Secret sharing distributes the ability to reconstruct a secret across multiple pieces. Encryption transforms a message using a key and is normally designed to provide confidentiality; authenticated-encryption modes can additionally provide integrity and authenticity. Steganography attempts to conceal the existence of a message within a cover object. These goals overlap in some applications but are not interchangeable [13]–[16]. AES is standardized as a symmetric block cipher [18], and NIST specifies modes such as CBC and GCM for different security services [19], [20]. A visual-share system should therefore be evaluated as a secret-sharing mechanism, not advertised as a universal replacement for authenticated encryption.

Literature Survey

The literature can be organized into five connected strands: foundational visual secret sharing; access structures and contrast optimization; grayscale, halftone, and meaningful-share extensions; text-oriented or non-image adaptations; and the surrounding cryptographic and information-hiding literature.

3.1 Foundational Visual Secret Sharing

Naor and Shamir introduced visual cryptography and formalized the idea that qualified subsets of transparencies can reveal a secret image through superposition [1]. Shamir's earlier threshold-secret-sharing work provides the broader mathematical context for distributing reconstruction capability among participants [2]. Ateniese et al. generalized visual cryptography to arbitrary access structures, allowing the qualified subsets to be defined beyond a simple threshold [4]. Blundo and De Santis examined constructions with perfect reconstruction of black pixels [6], while Blundo, De Santis, and Stinson analyzed contrast bounds and trade-offs [5].

3.2 Improving Contrast, Pixel Expansion, and Share Meaningfulness

A recurring challenge is that reconstructed images can lose contrast or require large shares. Work on contrast-optimal schemes and pixel-expansion bounds addresses the efficiency and legibility of reconstructed secrets [5], [9], [10]. Extended visual cryptography attempts to produce shares that contain meaningful cover images while still revealing the secret after stacking [11]. This is relevant to T2VS because binary arrays converted into image files may be visually uninformative and may be detectable as generated artifacts.

3.3 Grayscale, Halftone, and Color Schemes

Blundo, De Santis, and Naor developed visual cryptography for grayscale images, demonstrating that the framework can represent more than binary black-and-white content [3]. Zhou et al. introduced halftone visual cryptography to create shares with improved visual quality and meaningful appearance [7]. Color visual cryptography and other extensions expand the design space but introduce additional representation, contrast, and computational trade-offs. These developments show that share format and human interpretability are important engineering dimensions, not merely cosmetic details.

3.4 Probabilistic and Fault-Tolerant Schemes

Probabilistic visual cryptography studies randomized constructions and their reconstruction properties [8]. Fault-tolerant schemes address noise, imperfect alignment, and transmission or printing defects [12]. Such work is directly relevant to a text-share protocol: if one bit in a share is altered, a naïve combining operation can corrupt the corresponding recovered symbol. Error detection and correction must be part of the protocol rather than an afterthought.

3.5 Text-Oriented Visual Cryptography and Information Hiding

Text-oriented approaches have explored alternatives to ordinary pixel-based shares. Lin et al. proposed a natural-language-letter-based visual cryptography scheme in which letters are used in the share construction [17]. The broader steganography literature distinguishes concealment from cryptographic confidentiality and studies the limits of hiding information in media [13]–[16]. Recent reviews emphasize that steganalysis must be considered alongside embedding or share-generation methods [27], [28].

3.6 Research Gap

The supplied dissertation identifies the need for a direct mapping from text bytes to visual shares without first rendering a full text image. The gap is not simply the absence of text-to-binary conversion; binary conversion is routine. The substantive research challenge is to define a text-specialized share construction that preserves the formal secrecy condition, supports robust reconstruction, and has measurable advantages over established authenticated encryption or conventional secret sharing. A useful contribution must prove the distributional property, test leakage empirically, and quantify the cost of two-share transport.

Research strand	Representative works	Relevance to this paper
Threshold VC foundations	Naor & Shamir [1]; Shamir [2]	Formal access structures and reconstruction threshold

General access structures and contrast	Ateniese et al. [4]; Blundo et al. [5], [6]; Hofmeister et al. [9]	Security definitions, contrast, pixel expansion
Grayscale / halftone / extended shares	Blundo et al. [3]; Zhou et al. [7]; Ateniese et al. [11]	Share quality and meaningful representations
Robust / probabilistic VC	Cimato et al. [8]; fault-tolerant VSS [12]	Randomization and resilience to errors
Text and information hiding	Lin et al. [17]; Anderson & Petitcolas [13]; Fridrich [15]	Text-specific shares and detection risk

Proposed T2VS Framework and Methodology

4.1 System Overview

The proposed framework contains a sender, a byte-encoding module, a visual-share encoder, two independently transported share objects, a receiver-side validator, a share combiner, and a text decoder. The system is intended for short messages and requires the receiver to obtain both valid shares. A protocol header should carry a version identifier, message identifier, encoding type, original byte length, share dimensions, and integrity metadata. Header fields that reveal sensitive operational details should be minimized or protected.

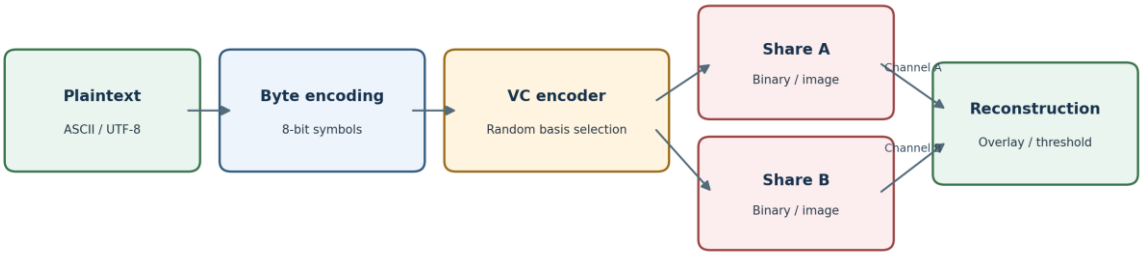

Fig. 1. *Conceptual T2VS sender–channel–receiver architecture.*

4.2 Source Prototype: Reconstructed Baseline

The dissertation describes the following baseline rule for each message bit b: if b = 0, both shares receive [0,0]; if b = 1, both shares receive [1,0] (or a randomized orientation [0,1] in parts of the description). The shares are concatenated into binary arrays, reshaped into images, and recombined with a bitwise OR operation. The implementation excerpt in the dissertation, however, uses the deterministic [1,0] pattern for every one bit, while other sections discuss random selection between [1,0] and [0,1]. This inconsistency should be resolved in any implementation.

Input bit	Share A pair	Share B pair	OR result
0	00	00	00
1 (implementation excerpt)	10	10	10
1 (random orientation variant)	10 or 01	same orientation in both shares	10 or 01

Under this rule, a single share reveals whether the corresponding bit was zero: the pair 00 occurs only for a zero bit, while 10/01 occurs only for a one bit. Therefore, the share distribution is secret-dependent. It is not information-theoretically private. This is a direct consequence of the rule, not a speculative attack. The dissertation itself discusses this leakage in its security-analysis chapter.

4.3 Corrected Basis-Matrix Design Direction

A defensible (2,2) visual-secret-sharing construction must select share patterns from basis matrices so that each individual share has the same marginal distribution for white and black secret pixels, while the combined shares have distinguishable Hamming weights. In a conventional illustrative construction with two subpixels, a white pixel uses identical row patterns across the two shares (one black subpixel after overlay), while a black pixel uses complementary patterns (two black subpixels after overlay). Randomly permuting the two columns prevents either participant's individual share from depending on the secret. The receiver decodes using the combined Hamming weight, not by treating the OR output as a literal zero/nonzero bit without threshold semantics [1], [5].

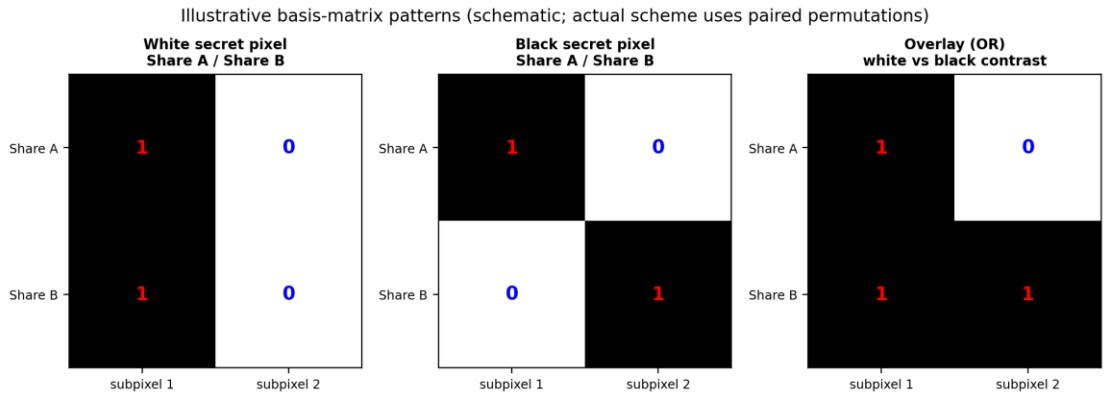

Fig. 2. *Schematic illustration of the basis-matrix principle. The exact matrices and polarity convention must be fixed in the implementation specification.*

The figure is conceptual and uses a simplified two-subpixel illustration. A publication-grade implementation should state the actual basis matrices, the permutation-selection procedure, the contrast threshold, and the formal distributional proof. The encoder must use a cryptographically secure random source where unpredictability is required. The same random permutation must not be reused in a way that creates cross-message correlation.

4.4 Encoding Procedure

1. Validate input and normalize it according to the protocol's character-encoding policy.
2. Encode the text as bytes; record the byte length and encoding identifier.
3. Optionally append a message authentication tag or integrity digest in a clearly specified hybrid construction.
4. Convert each byte into a fixed-width bit sequence.
5. For each bit, select the appropriate basis-matrix column pattern using independent cryptographically secure randomness.
6. Append the generated subpixels to Share A and Share B in a deterministic, documented order.
7. Serialize each share with version, dimensions, message identifier, and integrity-protected metadata.
8. Transmit the two shares over separately managed channels and avoid exposing a common identifier if correlation is a concern.

4.5 Decoding Procedure

9. Receive both share objects and verify protocol version, message identifier, dimensions, and expected payload length.
10. Authenticate the share containers and reject malformed or unauthenticated inputs.
11. Align corresponding subpixel groups; if image transport is used, apply registration and geometric correction before decoding.
12. Combine corresponding subpixels using the scheme-defined operation.
13. Measure the combined Hamming weight for each encoded bit and apply the specified threshold rule.
14. Reassemble the recovered bits into bytes, verify the integrity tag, and decode using the declared character encoding.
15. Return the recovered message only if all structural and integrity checks succeed; otherwise request retransmission.

4.6 Algorithmic Complexity

Let L be the number of input bytes and let m be the pixel expansion per source bit. The source contains 8L message bits. A direct encoder and decoder therefore require O(8Lm) time and O(8Lm) storage for the share arrays. With fixed m, both are linear in message length: O(L). The constant factor depends on matrix selection, random-number generation, image serialization, integrity protection, and transport—not merely on the OR operation. Any timing comparison must measure the same end-to-end stages for all methods.

4.7 Threat Model and Security Objectives

Threat / objective	Required property	Design implication
Single-share interception	No information about secret under formal forbidden-set definition	Use basis matrices with secret-independent marginal share distributions
Both-share compromise	Confidentiality may be lost	Protect endpoint storage and transport; minimize retention
Tampering / substitution	Integrity and authenticity	MAC or authenticated encryption around share containers
Reordering / replay	Freshness and correct pairing	Unique message IDs, sequence numbers, authenticated metadata
Transmission noise	Reliable reconstruction	Error detection, FEC, retransmission, alignment checks
Traffic analysis	Reduce metadata leakage	Channel diversity, timing/size analysis, padding where justified
Statistical steganalysis	Avoid detectable share biases	Randomness tests and trained detection evaluation

4.8 Security Analysis: Essential Corrections

The baseline deterministic mapping is not a secure (2,2) secret-sharing construction because the share's symbol distribution reveals each plaintext bit. Calling a share 'random-looking' does not establish perfect secrecy. The corrected design must demonstrate, for each forbidden share set S and any two secrets x and x′, that Pr[S=s | X=x] = Pr[S=s | X=x′] for every possible share observation s.

Equivalently, the mutual information I(X;S) should be zero under the stated model. This condition must be verified for the exact matrices and implementation.

Visual cryptography alone does not automatically provide message authentication, sender identity, replay protection, availability, or secure channel establishment. A malicious intermediary can replace or corrupt a share unless the container is authenticated. In practice, a hybrid architecture should use established authenticated encryption for data-in-transit or authenticate each share object with a secure MAC/signature. The system should not claim that removing key management is an unconditional advantage: a share-based design moves the trust requirement to share distribution, storage, pairing, and channel independence.

Experimental Setup and Source-Reported Results

5.1 Dataset and Environment in the Dissertation

The supplied dissertation describes a synthetic dataset of 500 short intelligence-style messages. It reports 200 short messages (10–30 characters), 200 medium messages (50–100 characters), and 100 long messages (150–200 characters), using printable ASCII characters 32–126. The reported environment is Windows 11 Pro, Python 3.9, Pillow, PyCryptodome, NumPy, and Pandas, on an Intel Core i5 system restricted to one core and a nominal 1.0 GHz cap, with 4 GB RAM and SSD storage.

Category	Length range	Messages	Example type
Short	10–30 characters	200	Alert / coordinate
Medium	50–100 characters	200	Description / instruction
Long	150–200 characters	100	Situation report

5.2 Reported Reconstruction and Timing

The dissertation reports that all 500 messages were reconstructed exactly. It reports the following average encoding times for T2VS and AES-128. These are transcribed from the source document and have not been independently rerun for this manuscript.

Message length	AES-128 average (ms)	T2VS average (ms)	Reported speed ratio
30 characters	2.12	0.11	19.3×
100 characters	5.84	0.36	16.2×
200 characters	11.42	0.72	15.9×

The dissertation also reports 100% exact reconstruction for plaintext, AES-128, and T2VS in the tested dataset. That result is a functional round-trip result under the test conditions; it is not evidence that T2VS provides confidentiality. The flawed deterministic mapping can reconstruct perfectly while leaking the plaintext bits from an individual share.

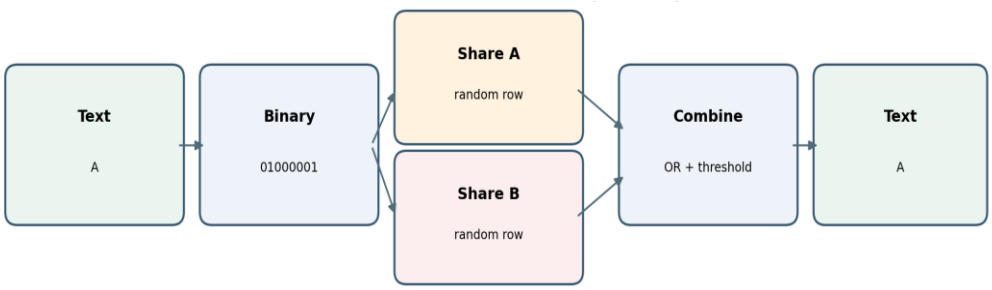

Fig. 3*. Conceptual text-to-share round trip showing message serialization, two-share generation, combination, and recovery.*

5.3 Interpretation and Reproducibility Caveats

The source-reported timing advantage should not be generalized without a controlled rerun. The baseline comparison uses AES-CBC with a shared key and IV, while a production comparison should include authenticated encryption such as AES-GCM or ChaCha20-Poly1305 and should account for key setup, nonce generation, tag size, image encoding, file I/O, and network transfer. AES may also be hardware accelerated on some processors. The dissertation itself notes that the comparison is not fully equivalent because AES and T2VS provide different security properties.

The source contains inconsistent descriptions of randomization and share generation. Some sections state that one bits use randomized orientations; the code excerpt uses deterministic [1,0] for one bits. The security discussion also recognizes that [0,0] positions disclose zero bits. Consequently, the reported reconstruction and timing numbers are useful as a baseline implementation record, but they cannot validate the confidentiality of the described rule. A corrected construction requires new experiments.

5.4 Recommended Evaluation Protocol

Metric	Measurement procedure	Expected reporting
Round-trip correctness	Exact byte-for-byte comparison over test corpus	Success rate, BER, failed cases

Forbidden-share leakage	Estimate I(X;S), train classifiers to infer bit/byte statistics	Leakage estimate, confidence intervals, baseline comparison
Randomness / detectability	NIST-style statistical tests where appropriate; local pattern tests; steganalysis classifier	Test statistics, false-positive and detection rates
Latency	Repeated warm/cold runs with fixed hardware and software	Median, mean, standard deviation, p95
Storage / transmission	Measure serialized shares plus headers, tags, and FEC	Bytes per plaintext byte and overhead
Channel robustness	Inject bit flips, packet loss, reordering, image scaling, and compression	BER, recovery rate, retransmission rate
Integrity	Tamper, replay, and share-substitution tests	Detection rate and false acceptance rate
Energy / memory	Measure on target embedded hardware	Joules/message, peak RAM, CPU time

Discussion

The principal attraction of a text-oriented visual-share framework is architectural simplicity: a message can be represented as bytes, transformed into shares, and reconstructed through a small set of deterministic operations. For short messages, the computational work is likely to be linear and modest. However, the system pays for this simplicity with duplicated payload, two-object management, synchronization requirements, and potentially reduced operational reliability when channels are not independent.

The most important finding of this critical review is that a correct implementation of the intended algorithm is not equivalent to a secure implementation. The baseline T2VS mapping demonstrates reversible encoding, but the deterministic zero pattern leaks information. A corrected basis-matrix design is therefore a prerequisite, not an optional enhancement. In addition, the security objective must be stated narrowly: a (2,2) secret-sharing mechanism protects against an adversary who obtains fewer than the threshold number of shares, provided the construction is formally correct and the shares are not correlated or compromised together. The design should also be compared against a realistic baseline. AES-GCM and ChaCha20-Poly1305 provide authenticated encryption with standardized security properties [18]–[23]. A T2VS prototype may be faster in a narrow software microbenchmark, but that does not imply it is more secure or more efficient end-to-end. The relevant comparison should include message confidentiality, integrity, authentication, share delivery, key-management burden, storage overhead, and threat assumptions.

7. Detailed Protocol Architecture and Data Representation

A research implementation should distinguish the logical message from its serialized share containers. The logical message is the byte sequence to be protected. The share container is a transport object that carries the generated binary pattern together with the minimum metadata needed for correct reconstruction. Keeping these layers separate makes it possible to change the image format or channel transport without changing the mathematical secret-sharing construction.

A recommended container schema includes: protocol version; algorithm identifier; character encoding identifier; message identifier; share index; threshold and participant count; original byte length; encoded bit length; subpixel expansion factor; image width and height (if rasterized); generation nonce or randomness context identifier; and integrity/authentication data. Fields that are not required by the receiver should be omitted to reduce metadata leakage. All metadata that affects interpretation must be authenticated. For binary payloads, a compact packed-bit representation is generally more efficient than storing one pixel per byte in an uncompressed grayscale image. If an image representation is required for operational or human-handling reasons, the implementation should specify pixel polarity, row-major ordering, bit packing, image dimensions, color mode, compression behavior, and whether lossy transformations are prohibited. Lossy formats such as JPEG should not be used for raw share patterns because quantization can change pixel values and therefore alter reconstruction.

Field	**Purpose**	**Validation rule**
version / algorithm_id	Select exact decoder and basis matrices	Reject unknown or downgraded versions
message_id / share_index	Pair Share A with Share B	Require matching message ID and distinct share indexes
encoding / original_length	Reconstruct bytes without ambiguity	Length must match decoded payload
expansion / dimensions	Map groups to source bits	Check dimensions and expected number of subpixels
integrity tag	Detect tampering and substitution	Verify before decoding or release
sequence / expiry	Replay and freshness control	Reject duplicates or expired containers

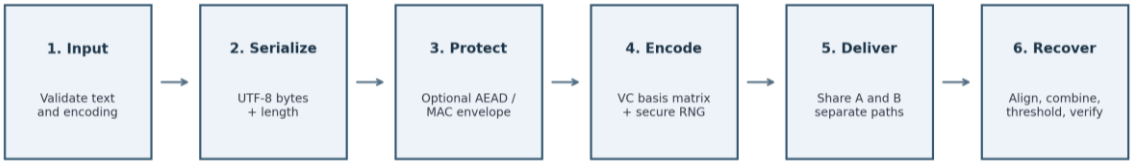

Fig. 4. *Expanded processing lifecycle from input validation through receiver-side verification.*

8. Mathematical Formulation of the (2,2) Construction

Let X be a binary secret pixel, where X=0 denotes white and X=1 denotes black. Let Share A and Share B each contain a two-subpixel row. A conventional illustrative (2,2) construction uses two possible column permutations for each secret value. For a white secret pixel, the permitted matrices are C0 = {[[1,0],[1,0]], [[0,1],[0,1]]}. For a black secret pixel, the permitted matrices are C1 = {[[1,0],[0,1]], [[0,1],[1,0]]}. Each matrix is selected with probability one half.

The row corresponding to Share A is either [1,0] or [0,1] with equal probability for both X=0 and X=1. The same is true for Share B. Therefore, the marginal distribution of either individual share is independent of X. In probability notation, for every possible individual share value s, Pr[Share A=s | X=0] = Pr[Share A=s | X=1], and likewise for Share B. This is the key forbidden-set secrecy condition for the (2,2) scheme.

When the shares are superimposed using bitwise OR, a white secret pixel produces a row with Hamming weight one, while a black secret pixel produces [1,1], with Hamming weight two. The receiver can therefore classify the combined subpixel pair by a threshold on its weight. The exact interpretation of 0/1 as white/black is a convention and must be consistent across the encoder, image writer, and decoder.

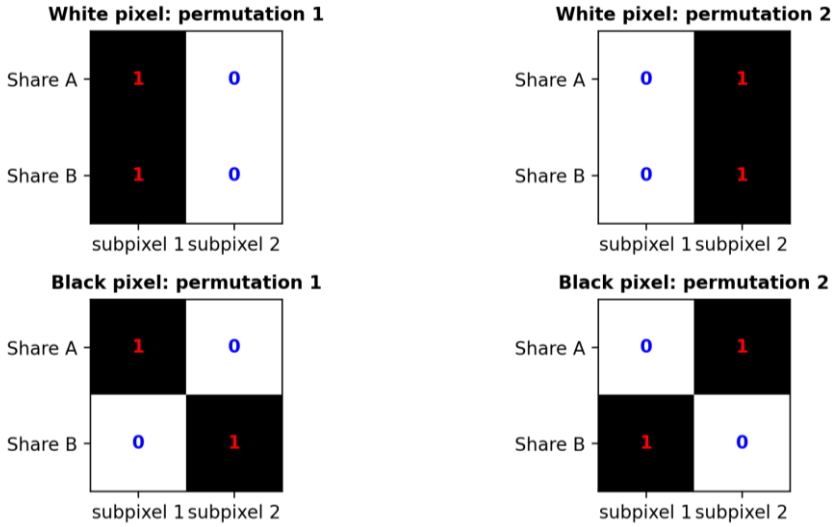

Fig. 5. *Basis-matrix sets for a standard illustrative (2,2) visual-secret-sharing construction.*

Property	**White secret pixel**	**Black secret pixel**
Possible matrices	C0 contains identical rows	C1 contains complementary rows
Share A marginal	10 or 01, each with probability 1/2	10 or 01, each with probability 1/2
Share B marginal	10 or 01, each with probability 1/2	10 or 01, each with probability 1/2
OR-combined row	10 or 01; weight = 1	11; weight = 2
Single-share leakage	No bit information under ideal model	No bit information under ideal model

This construction is an illustrative basis-matrix explanation, not a claim that the supplied dissertation's implementation already uses these matrices. The source code excerpt instead uses a deterministic pair for zero bits and a deterministic pair for one bits. Replacing that code with the basis-matrix construction changes the algorithm and requires a fresh experimental evaluation.

9. Encoding and Decoding Algorithms: Implementation-Level Specification

The encoder should be specified as a deterministic protocol around a randomized share-generation core. Given a byte string P, the system first computes B=8|P| bits. For each source bit b_i, it samples an independent uniform permutation π_i from the two-element permutation set and applies π_i to the appropriate basis matrix C_bi. The first row is appended to Share A and the second row to Share B. The resulting shares contain 2B subpixels each. If represented as packed binary arrays, each share requires 2B bits, before metadata and authentication overhead.

The decoder accepts two authenticated containers. It verifies the message identifier, share indexes, version, length, and dimensions. It then groups each share into pairs, OR-combines corresponding subpixels, and classifies each pair using the specified Hamming-weight rule: weight one maps to a white source bit and weight two maps to a black source bit. It then packs the recovered bits into bytes and decodes them using the declared encoding. Any invalid weight, length mismatch, authentication failure, or decoding error causes rejection rather than silent output.

Algorithm stage	**Input**	**Operation**	**Output**
Input preparation	Text M	Encode to declared byte encoding	P, byte length L
Bit serialization	P	Expand each byte to eight bits	B, length 8L
Share generation	Each bit b_i	Select random permutation and basis matrix C_bi	Two subpixel rows

Container creation	Share arrays + metadata	Serialize, authenticate, and optionally pad	Share A and Share B
Share validation	Two containers	Check authenticity, pairing, dimensions, version	Validated shares
Combination	Aligned share rows	Bitwise OR and Hamming-weight threshold	Recovered bits
Text recovery	Recovered bits	Pack bytes and decode declared encoding	Recovered message M′

The implementation should use a cryptographically secure random-number generator rather than a conventional simulation PRNG for the basis-matrix permutation. Test-only deterministic seeds may be used for reproducible test vectors, but must never be reused in production share generation. Randomness failure should cause the encoder to stop, not silently fall back to a predictable generator.

10. Complexity, Storage Overhead, and Scalability

Let L denote the message length in bytes, B=8L the number of source bits, and m=2 the subpixel expansion per source bit in each share for the illustrative construction. Each share contains 2B=16L binary subpixels. The two shares together contain 4B=32L binary subpixels, equivalent to 4L bytes when bit-packed. This is a fourfold aggregate payload expansion relative to the original byte sequence, excluding metadata, authentication tags, file headers, padding, and error-correction redundancy. Each individual share has a payload size of approximately 2L bytes when bit-packed.

The encoder processes a constant number of operations per source bit: one random permutation selection, a small matrix lookup, and two subpixel writes. Its time complexity is O(L). The decoder similarly performs one OR and one weight classification per source bit, so its computational complexity is O(L). The storage complexity is O(L) because both share arrays scale linearly with the number of input bytes. Image encoding, serialization, cryptographic authentication, network I/O, and error correction may dominate the cost in practical systems.

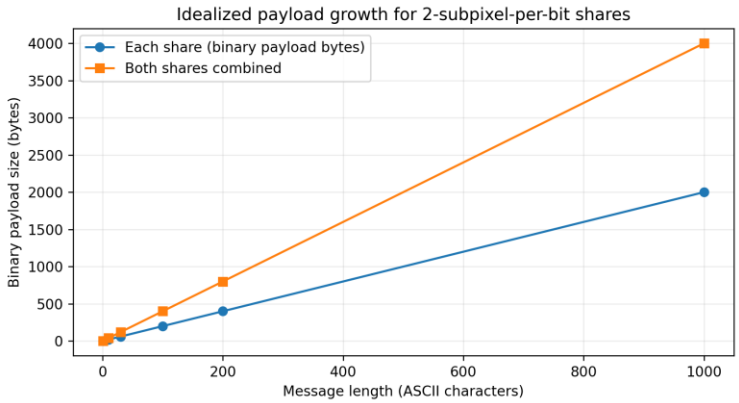

Fig. 6. *Idealized bit-packed payload growth for the illustrative two-subpixel-per-bit construction; excludes all protocol overhead.*

Message size	**Each share payload**	**Both shares combined**	**Interpretation**
30 bytes	60 bytes	120 bytes	4× aggregate binary payload
100 bytes	200 bytes	400 bytes	Before headers and integrity fields
200 bytes	400 bytes	800 bytes	Matches the small-message use case
1,000 bytes	2,000 bytes	4,000 bytes	Linear growth; transport overhead becomes relevant

11. Dataset Design, Sampling, and Experimental Controls

The dissertation describes a synthetic corpus rather than a publicly released operational dataset. Synthetic messages are appropriate for functional testing because they avoid the disclosure of sensitive operational content and allow message length and character distribution to be controlled. They do not, however, establish performance on all natural-language messages, multilingual content, binary payloads, or adversarially chosen inputs.

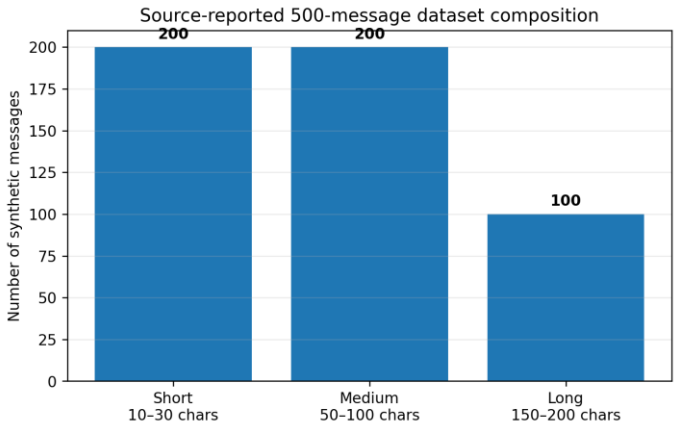

Fig. 7. *Distribution of the 500 synthetic messages described in the dissertation.*

The source reports three length groups: 200 messages of 10–30 characters, 200 messages of 50–100 characters, and 100 messages of 150–200 characters. It also reports printable ASCII codes 32–126, with an intended character mix of approximately 60% alphabetic characters, 20% numeric characters, 15% spaces, and 5% punctuation or symbols. The source's dataset description should be treated as the experimental specification; the actual dataset files and generation seed should be released to permit independent verification.

Control dimension	Recommended control	Reason
Message corpus	Publish generated messages or a deterministic generator and seed	Enables independent reproduction
Hardware	Report CPU model, core policy, clock policy, RAM, OS	Timing depends on processor and power state
Software	Pin Python and library versions; publish lockfile	Avoids implementation drift
Timing	Use monotonic high-resolution timer; warm-up; repeat runs	Reduces measurement noise
I/O	Report computation-only and end-to-end timings separately	Prevents excluding operational cost
Randomness	Use secure randomness in production; fixed test vectors separately	Reproducibility without weakening deployment
Baseline	Include AES-GCM and ChaCha20-Poly1305 in addition to legacy CBC comparison	Compare with authenticated encryption

12. Detailed Interpretation of Source-Reported Performance

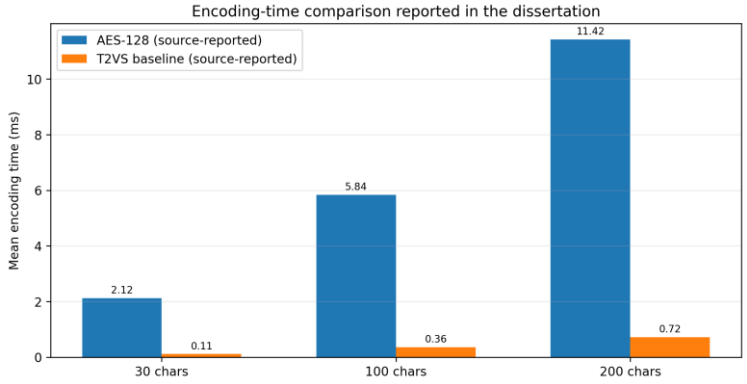

Fig. 8. *Encoding times transcribed from the dissertation; they are not independently reproduced in this manuscript.*

The source reports mean encoding times of 2.12 ms for AES-128 and 0.11 ms for T2VS at 30 characters; 5.84 ms and 0.36 ms respectively at 100 characters; and 11.42 ms and 0.72 ms respectively at 200 characters. These correspond to reported ratios of approximately 19.3×, 16.2×, and 15.9×. The trend is consistent with both algorithms scaling approximately linearly over the measured message sizes, but the comparison cannot establish that T2VS is generally faster than modern cryptography.

The AES baseline in the dissertation is AES-128 in CBC mode with PKCS#7 padding and a random IV. CBC provides confidentiality when used correctly but does not itself authenticate ciphertext. A contemporary comparison should include an authenticated-encryption mode such as AES-GCM, which NIST specifies as authenticated encryption with associated data [20], and ChaCha20-Poly1305 as specified in RFC 8439 [23]. The comparison must include key/nonce handling, tag generation, serialization, and file/network transfer.

The source reports decryption as 0.00 ms for T2VS because the combining operation is described as effectively instantaneous. A measured zero is usually a timer-resolution or reporting-floor issue, not evidence that the operation consumes no time. Future measurements should report a lower bound or nanoseconds per byte, and should time the full receiver pipeline including image loading, alignment, authentication, thresholding, byte packing, and text decoding.

Message length	AES-128 enc. (ms)	T2VS enc. (ms)	Reported ratio	Interpretation
30 chars	2.12	0.11	19.3×	Source-reported prototype result
100 chars	5.84	0.36	16.2×	Source-reported prototype result
200 chars	11.42	0.72	15.9×	Source-reported prototype result

The reported accuracy result is 500/500 successful reconstructions for plaintext, AES-128, and T2VS. It confirms that the tested pipelines returned the expected messages in the source's ideal test setup. It does not measure confidentiality, secrecy of one share, authenticity, resistance to statistical inference, or robustness under channel corruption. Functional correctness and cryptographic security are separate evaluation axes.

13. Statistical Analysis and Reporting Plan

For each message-length category and algorithm, the experiment should report the number of trials, mean, median, standard deviation, minimum, maximum, and 95th percentile. The timing distribution should be inspected for skew and outliers. Paired measurements on the same messages and machine state are preferable when comparing methods. Confidence intervals can be estimated using bootstrap resampling when timing distributions are non-normal. A speedup ratio should be reported with a confidence interval rather than as a single point estimate.

For reconstruction, report exact message success rate and bit error rate (BER). BER is the number of incorrect recovered bits divided by the total number of transmitted source bits. For confidentiality, report mutual-information estimates or classifier performance for an attacker given one share. A classifier that predicts source bits substantially better than chance indicates leakage; however, failure to train a classifier is not a proof of perfect secrecy. The formal distributional argument remains primary.

For steganalysis, report detection AUC, true-positive rate at fixed false-positive rates, and performance on held-out share-generation seeds. Evaluate both generic image detectors and scheme-aware tests. Randomness-test batteries can identify certain statistical anomalies but do not prove cryptographic security. Results must include the sample size, test parameters, and multiple-testing treatment.

14. Security Properties and Non-Properties



Fig. 9. *Security layers: threshold secrecy, reconstruction, and protocol-level protection are distinct.*

Security property	Provided by correct (2,2) VC alone?	Additional mechanism
Confidentiality against one share	Yes, if the forbidden-share distribution is secret-independent	Formal proof and correct implementation
Confidentiality if both shares compromised	No	Endpoint security or additional encryption
Integrity / tamper detection	No	MAC, digital signature, or AEAD
Sender authentication	No	Authenticated channel or signature
Replay protection	No	Sequence number, nonce, expiry, replay cache
Availability	No	Redundant channels, retransmission, recovery
Traffic-analysis resistance	No	Operational controls and metadata minimization
Steganographic undetectability	Not guaranteed	Separate covertness design and adversarial testing

The security analysis must also consider the adversary's observation model. An adversary may see one share, both shares, the channel metadata, or the endpoint memory. The classical (2,2) threshold guarantee only concerns a forbidden set containing fewer than two shares under the construction's assumptions. If both shares are transmitted through the same account, device, cloud bucket, or network path, the practical security may be weaker than the mathematical threshold suggests.

A further issue is share correlation. If random choices are reused across messages, or if the share-generation randomness is predictable, multiple observations may reveal relationships that are absent in a single-message proof. Randomness should be independently sampled at the required granularity. The implementation should be tested for accidental deterministic patterns, duplicate shares, repeated seeds, and leakage through headers or image dimensions.

15. Integrity, Authentication, and Hybrid Protection

A share-based scheme should not be mistaken for an authenticated communication protocol. An attacker who can modify a share may alter the reconstructed message or cause decoding failure. A robust deployment can authenticate each share container using a MAC, digital signature, or an authenticated-encryption wrapper. The choice depends on whether sender identity, non-repudiation, or shared-secret authentication is required.

A practical hybrid design can first authenticate/encrypt the plaintext using a standard AEAD construction, then secret-share the resulting ciphertext or an appropriately protected payload. This provides cryptographic confidentiality and integrity while using share distribution as an additional access-control layer. Such a design no longer claims to eliminate key management; instead, it explicitly combines standard cryptographic protection with threshold distribution. The trade-off should be evaluated transparently. If the goal is to avoid a shared symmetric key, digital signatures can authenticate the sender but do not by themselves provide confidentiality. If the goal is to protect against one channel being compromised, secret sharing may be useful, but both shares must be protected against correlation, replay, and substitution. These are protocol-design decisions that should be specified before implementation.

16. Robustness and Error-Correction Strategy

The original prototype assumes error-free transmission. Real communication paths may introduce packet loss, image resizing, compression, thresholding errors, channel reordering, and partial file corruption. A one-bit error in a share can change the OR-combined result, and a corrupted subpixel can cause an incorrect source bit. Therefore, the receiver should validate the container before reconstruction and should not silently guess missing data.

A CRC can detect accidental corruption but is not a cryptographic integrity mechanism. A MAC or AEAD tag is required against active attackers. Forward error correction can recover from a bounded level of channel noise, but it introduces redundancy and changes the payload size. Reed–Solomon codes are a well-established family of error-correcting codes [21]. An implementation should compare no FEC, simple repetition, and a suitable block code under controlled bit-flip and erasure rates.

Channel impairment	Potential consequence	Recommended handling
Single subpixel flip	Incorrect combined weight	Detect via authentication/checksum; FEC if appropriate
Lost packet / truncated file	Incomplete share	Length validation and retransmission
Share reordering	Wrong pairing	Authenticated message identifier and share index
Image scaling / resampling	Pixel values and alignment altered	Lossless format, dimensions, registration, threshold validation
JPEG compression	Subpixel corruption	Prohibit lossy image compression
Replay of old share	Stale message reconstruction	Sequence number, nonce, expiry, replay cache

17. Implementation Blueprint and Reproducibility Package

A research-grade implementation should be organized into independent modules: text serialization; basis-matrix construction; secure random permutation selection; share-array generation; container serialization; authentication; transport adapter; receiver validation; share alignment; threshold decoding; and experiment instrumentation. Unit tests should validate each module independently, while integration tests should verify full round trips.

Module	Core responsibility	Minimum test cases
serializer.py	UTF-8/ASCII bytes, length, normalization	Empty, ASCII, multibyte UTF-8, malformed input
basis.py	Define C0/C1 and allowed permutations	Exhaustive matrix membership and marginal distributions
encoder.py	Generate two shares	All-zero, all-one, alternating, random messages
container.py	Pack bits and metadata	Round-trip serialization, truncation, version mismatch
auth.py	MAC/AEAD/signature verification	Tamper, wrong key, replay, altered metadata
decoder.py	Pair, OR, threshold, byte reconstruction	Exact round-trip, invalid Hamming weights
benchmark.py	Timing, memory, size, energy hooks	Repeatability, timer resolution, CSV output
attacks.py	Single-share leakage and tamper tests	Classifier baseline, pattern-frequency tests

A reproducibility package should include source code, a requirements lockfile, compiler/runtime details, test vectors, synthetic dataset generator, dataset seed, raw timing logs, benchmark scripts, and an explanation of the exact basis matrices. The package

should include a security statement that identifies the construction version and its assumptions. A result should not be described as independently reproduced unless another researcher can execute the package and obtain comparable measurements.

The software should maintain separate test and production randomness paths. A deterministic random seed is useful for unit tests because it allows exact expected outputs. Production mode must use a cryptographically secure random source and should make it impossible to accidentally activate a deterministic seed. Logging should not record plaintext messages, secret random values, or complete shares unless the experiment explicitly requires them and the storage is protected.

18. Extended Comparative Evaluation

Dimension	**T2VS / visual secret sharing**	**AES-GCM / ChaCha20-Poly1305**	**Conventional steganography**
Primary goal	Distribute reconstruction capability across shares	Confidentiality and integrity of data	Conceal existence of communication
Key requirement	No conventional decryption key in pure VSS; shares must be protected and paired	Secret key and nonce management	Depends on embedding method; often key or shared parameters
Number of transmitted objects	At least two in (2,2) scheme	Usually one ciphertext object	One cover object per embedded message
Integrity	Not inherent	Built into AEAD construction	Not inherent unless separately added
Single-object leakage	Must be zero under forbidden-set proof	Ciphertext should reveal no plaintext beyond allowed leakage	Depends on cover and embedding model
Traffic metadata	Two-share exchange may be conspicuous	Encrypted traffic may be identifiable	Goal is to hide communication existence, but detection remains possible
Payload overhead	Pixel expansion and duplicated shares	Nonce/tag/padding overhead	Limited by cover capacity and embedding distortion
Best-fit role	Threshold access to a visual/binary secret	General-purpose secure data transport	Covert data embedding in cover media

This comparison is qualitative. It does not establish universal superiority for any technique. A fair benchmark must compare equivalent security goals and threat models. In particular, pure visual secret sharing and authenticated encryption solve different problems. If the application requires secure network communication, AEAD should be treated as a strong baseline. If the application requires distributing authority among multiple holders, threshold secret sharing is the relevant conceptual comparison.

19. Limitations and Threats to Validity

Construct validity is a concern because encryption time alone is not an adequate measure of secure communication performance. A T2VS encoding microbenchmark may omit image serialization, file storage, channel transfer, pairing, authentication, and decoding. A complete evaluation must measure both algorithmic and operational costs.

Internal validity is limited by the reported single-machine setup and the possibility of processor-specific acceleration. AES implementations may use hardware instructions, while Python-level loops may affect the T2VS implementation differently. The exact software versions, CPU frequency behavior, timer resolution, and process scheduling can influence the measured ratio.

External validity is limited by the synthetic dataset, printable ASCII restriction, and short message lengths. Results may not generalize to multilingual text, binary data, long reports, mobile devices, low-bandwidth radio links, or real adversarial environments. A second dataset and multiple hardware platforms are needed.

Conclusion validity is limited by the small number of reported benchmark categories and the absence of raw per-message measurements in the material available for this manuscript. Means without confidence intervals cannot establish statistical significance. Security validity is more serious: the deterministic baseline mapping violates the single-share secrecy condition, so its reported functional success cannot be interpreted as evidence of secure confidentiality.

A further limitation is the distinction between a conceptual corrected scheme and a completed implementation. The basis-matrix design described here is a recommended redesign. It must be implemented, tested against formal properties, and benchmarked independently before the manuscript can claim the corrected T2VS system as an experimentally validated contribution.

20. Future Research Roadmap

Research phase	**Work package**	**Expected output**	**Evaluation gate**
Phase I: Formal redesign	Specify matrices, permutation distribution, decoding threshold	Complete algorithm specification and proof	Forbidden-share distribution independent of secret

Phase II: Prototype	Implement encoder/decoder and container format	Reproducible reference implementation	Unit and property-based tests pass
Phase III: Security testing	Single-share inference, statistical tests, tamper/replay attacks	Security evaluation report	No detectable secret leakage under stated model
Phase IV: Reliability	Noise, erasure, misalignment, FEC experiments	Robust decoder and measured recovery curve	Defined error tolerance and failure behavior
Phase V: Benchmarking	Compare against AES-GCM and ChaCha20-Poly1305	Latency, size, memory, energy results	Equivalent end-to-end benchmark
Phase VI: Deployment study	Mobile/edge hardware and independent channels	Prototype deployment evaluation	Operational feasibility and threat-model review

Future research may also investigate general access structures such as (k,n) schemes, where multiple organizations or agents jointly authorize reconstruction. Another direction is meaningful-share generation, where the shares resemble ordinary documents or natural-language content. The letter-based visual cryptography work by Lin et al. [17] provides a relevant point of comparison. However, meaningful appearance must not weaken the formal secrecy condition or introduce detectable statistical artefact's.

A further research direction is adaptive share encoding. Machine-learning methods could be used to characterize the detectability of share patterns under a specific cover medium or communication channel. Such methods should be used as evaluation tools rather than as a substitute for a security proof. Any learned transformation must preserve the marginal distribution condition and must be tested against adaptive adversaries.

Finally, deployment research should evaluate whether two independent communication channels are genuinely available in the target environment. If both shares are stored on the same device or delivered through the same provider, the assumed threshold separation may not hold operationally. The channel-independence assumption should therefore be measured and documented rather than merely stated.

21. Conclusion and Research Implications

The expanded analysis supports three conclusions. First, direct text-to-share conversion is a feasible engineering approach in terms of asymptotic complexity: a fixed-expansion visual-secret-sharing encoder and decoder can operate in linear time with message length. Second, functional reconstruction and cryptographic secrecy are separate requirements. The source dissertation's baseline pair mapping can reconstruct the message exactly while exposing information in an individual share. Third, a corrected visual-secret-sharing construction must be defined using basis matrices whose forbidden-share distributions are independent of the secret, followed by a complete protocol that addresses authentication, replay, errors, metadata, and channel separation.

The source dissertation reports a 500-message synthetic dataset, 100% reconstruction in its test setup, and low encoding times for short messages. Those values provide a useful baseline record. They do not establish the security or operational superiority of the proposed approach. A new implementation based on a formal (2,2) basis-matrix construction must be evaluated from the beginning, with authenticated-encryption baselines and end-to-end transport measurements.

Accordingly, the research contribution should be framed as a critical and extensible text-oriented visual-secret-sharing framework. Its novelty claim should focus on the protocol adaptation, formal validation, robustness, and deployment evaluation that are actually demonstrated. This framing is more scientifically defensible than claiming that a simple binary expansion is inherently secure or universally faster than modern cryptography.

References

[1] M. Naor and A. Shamir, "Visual cryptography," in Advances in Cryptology—EUROCRYPT '94, Lecture Notes in Computer Science, vol. 950, Springer, 1995, pp. 1–12. doi: 10.1007/BFb0053419.

[2] A. Shamir, "How to share a secret," Communications of the ACM, vol. 22, no. 11, pp. 612–613, 1979. doi: 10.1145/359168.359176.

[3] C. Blundo, A. De Santis, and M. Naor, "Visual cryptography for grey level images," Information Processing Letters, vol. 75, no. 6, pp. 255–259, 2000. doi: 10.1016/S0020-0190(00)00108-3.

[4] G. Ateniese, C. Blundo, A. De Santis, and D. R. Stinson, "Visual cryptography for general access structures," Information and Computation, vol. 129, no. 2, pp. 86–106, 1996. doi: 10.1006/inco.1996.0076.

[5] C. Blundo, A. De Santis, and D. R. Stinson, "On the contrast in visual cryptography schemes," Journal of Cryptology, vol. 12, pp. 261–289, 1999. doi: 10.1007/s001459900057.

[6] C. Blundo and A. De Santis, "Visual cryptography schemes with perfect reconstruction of black pixels," Computers & Graphics, vol. 22, no. 4, pp. 449–455, 1998. doi: 10.1016/S0097-8493(98)00034-X.

[7] Z. Zhou, G. R. Arce, and G. Di Crescenzo, "Halftone visual cryptography," IEEE Transactions on Image Processing, vol. 15, no. 8, pp. 2441–2453, 2006. doi: 10.1109/TIP.2006.875249.

[8] S. Cimato, R. De Prisco, and A. De Santis, "Probabilistic visual cryptography schemes," The Computer Journal, vol. 49, no. 1, pp. 97–107, 2006. doi: 10.1093/comjnl/bxh152.

[9] T. Hofmeister, M. Krause, and H. U. Simon, "Contrast-optimal k out of n secret sharing schemes in visual cryptography," Theoretical Computer Science, vol. 240, no. 2, pp. 471–485, 2000. doi: 10.1016/S0304-3975(99)00243-1.
[10] W. G. Tzeng and C. M. Hu, "A new approach for visual cryptography," Designs, Codes and Cryptography, vol. 27, pp. 207–227, 2002.
[11] G. Ateniese, C. Blundo, A. De Santis, and D. R. Stinson, "Extended capabilities for visual cryptography," Theoretical Computer Science, vol. 250, nos. 1–2, pp. 143–161, 2001. doi: 10.1016/S0304-3975(99)00127-9.
[12] J. S.-T. Juan, Y.-C. Chen, and S. Guo, "Fault-tolerant visual secret sharing schemes without pixel expansion," Applied Sciences, vol. 6, no. 1, art. 18, 2016. doi: 10.3390/app6010018.
[13] R. J. Anderson and F. A. P. Petitcolas, "On the limits of steganography," IEEE Journal on Selected Areas in Communications, vol. 16, no. 4, pp. 474–481, 1998.
[14] S. Katzenbeisser and F. A. P. Petitcolas, Eds., Information Hiding Techniques for Steganography and Digital Watermarking. Artech House, 2000.
[15] J. Fridrich, Steganography in Digital Media: Principles, Algorithms, and Applications. Cambridge University Press, 2009.
[16] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker, Digital Watermarking and Steganography, 2nd ed. Morgan Kaufmann, 2008.
[17] H.-C. Lin, C.-N. Yang, C.-S. Laih, and H.-T. Lin, "Natural language letter based visual cryptography scheme," Journal of Visual Communication and Image Representation, vol. 24, no. 3, pp. 318–331, 2013. doi: 10.1016/j.jvcir.2013.01.003.
[18] National Institute of Standards and Technology, Advanced Encryption Standard (AES), FIPS PUB 197, upd. 1, 2023. doi: 10.6028/NIST.FIPS.197-upd1.
[19] M. Dworkin, Recommendation for Block Cipher Modes of Operation: Methods and Techniques, NIST SP 800-38A, 2001. doi: 10.6028/NIST.SP.800-38A.
[20] M. Dworkin, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST SP 800-38D, 2007. doi: 10.6028/NIST.SP.800-38D.
[21] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," Journal of the Society for Industrial and Applied Mathematics, vol. 8, no. 2, pp. 300–304, 1960.
[22] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ forward error correction scheme for object delivery," RFC 6330, IETF, 2011.
[23] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF protocols," RFC 8439, IETF, 2018. doi: 10.17487/RFC8439.
[24] F. Yergeau, "UTF-8, a transformation format of ISO 10646," RFC 3629, IETF, 2003. doi: 10.17487/RFC3629.
[25] C. E. Shannon, "Communication theory of secrecy systems," Bell System Technical Journal, vol. 28, no. 4, pp. 656–715, 1949. doi: 10.1002/j.1538-7305.1949.tb00928.x.
[26] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.
[27] P. Punithavathi and S. Geetha, "Visual cryptography: A brief survey," Information Security Journal: A Global Perspective, vol. 26, no. 6, pp. 305–317, 2017. doi: 10.1080/19393555.2017.1386249.
[28] M. H. Kombrink, Z. J. M. H. Geradts, and M. Worring, "Image steganography approaches and their detection strategies: A survey," ACM Computing Surveys, vol. 57, no. 2, art. 33, 2024. doi: 10.1145/3694965.
[29] P. M. Bachiphale and N. S. Zulpe, "A comprehensive review of visual cryptography for enhancing high-security applications," Multimedia Tools and Applications, vol. 84, pp. 31023–31045, 2025. doi: 10.1007/s11042-024-20426-1.