

Article

Structural Cryptanalysis of a Middle-Graph Matrix Cipher and a Canonical Public Middle-Graph Encoding

Farkhanda Yasmin ¹, Saima Nazeer ¹, Rehana Ashraf ¹, Muhammad Saeed Ahmad ² 
and Muhammad Waseem Asghar ^{3,*} 

¹ Department of Mathematics, Lahore College for Women University, Lahore 54000, Pakistan; farkhandayasmin01@gmail.com (F.Y.); saima.nazeer@lcwu.edu.pk (S.N.); rehana.ashraf@lcwu.edu.pk (R.A.)

² Department of Mathematics, University of Management and Technology, Lahore 54000, Pakistan; saeedkhan099@gmail.com

³ Department of Mechanical Engineering Science, University of Johannesburg, Johannesburg 2092, South Africa

* Correspondence: waseema@uj.ac.za

Abstract

Structural complexity in a graph-derived representation does not by itself establish cryptographic security. This paper performs a self-cryptanalysis of the authors' own middle-graph matrix cipher from an earlier version of this manuscript; the attacked construction is not attributed to a previously published cipher. For a fixed key, the cipher right-multiplies a normalized weighted adjacency matrix of $M(S_{n+1})$ by an invertible secret matrix. We prove a ciphertext-visible row relation that, from one exact ciphertext, recovers every leaf parameter y_i and the n secret key rows indexed by the edge vertices. A single ciphertext does not by itself reveal the message coordinates x_i or z_i ; complete key recovery requires additional information such as known plaintexts and a full-column-rank stacked system. We give a concrete recovery algorithm and numerical example, an $O(n^3)$ arithmetic cost analysis, a perturbation bound for rounded ciphertexts, and experiments showing when integer parameter recovery survives decimal rounding. As a representation-only alternative, we define a canonical public byte-to-residue encoding and an optional expanded middle-graph matrix over $\mathbb{F}_{65521}$. The compact residue vector is the primary serialized object; the dense graph matrix is reconstructed only when a downstream graph or matrix algorithm actually consumes it. The map is public, deterministic, and reversible and is not claimed to provide confidentiality. Its clique identities can detect non-adversarial random corruption under an explicitly stated model, but they are not a form of authentication. If protected transport is required, then a standard AEAD scheme is applied as an independent security layer. This composition is included only to delimit the security boundary and not as a new cryptographic primitive. Benchmarks report the medians and interquartile ranges, compare the graph encoder with a simpler 16-bit baseline, and compare direct AES-GCM with encoding-plus-AES-GCM. The results show that the graph representation is justified only when its explicit graph structure is needed downstream; otherwise, direct authenticated encryption is simpler and faster.

Keywords: middle graph; structural cryptanalysis; finite-field encoding; authenticated encryption; canonical indexing; adjacency matrix; exact arithmetic

MSC: 05C76; 05C50; 68P25; 94A60; 11T71



Academic Editor: Víctor Casamayor-Pujol

Received: 2 September 2026

Revised: 2 October 2026

Accepted: 6 October 2026

Published: 11 October 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The design of a cryptographic system requires more than a large collection of possible keys or a visually complicated transformation. Goldreich, Stinson and Paterson, and Katz and Lindell provided the standard cryptographic viewpoint in which a reversible representation is distinguished from encryption and security is evaluated with respect to explicit adversarial capabilities and success criteria [1–3]. This viewpoint is directly relevant to the present investigation because the candidate construction combines a public graph-derived representation with a secret right multiplication. The central question is therefore not whether the resulting matrix appears complicated but whether the secret multiplication actually prevents an adversary from recovering plaintext-dependent or key-dependent information.

The attacked construction should be identified unambiguously. It is the authors' own candidate design from an earlier version of this manuscript and not a cipher copied from or attributed to the authors of [4–7]. No separate published specification exists. We retain it here because a self-cryptanalysis is scientifically useful when the target is specified completely and the negative result is stated without transferring the weakness to unrelated graph-based schemes. Recent work also illustrates why such scrutiny matters: graph-based encryption proposals continue to appear, while dedicated cryptanalysis can expose structural weaknesses that are not visible from examples or output statistics alone [8,9].

The graph-theoretic literature supplies the structural language used to answer that question. Standard treatments of algebraic and general graph theory establish adjacency matrices, graph transformations, and related algebraic representations [10,11], while Priyadarsini surveyed ways in which graph-theoretic concepts have been incorporated into cryptographic algorithms [12]. Related work on line graphs and structured networks shows that derived graphs can possess highly organized, exactly computable descriptors, including M-polynomials and degree-based topological indices [13]. Recent studies further illustrate this algebraic viewpoint through degree-based graph invariants and adjacency matrix spectral quantities [14,15]. These studies are relevant here because they emphasize that apparently elaborate graph structures still contain deterministic algebraic information. In the present paper, that observation is applied in a cryptanalytic rather than a descriptor-setting context. The support pattern and row relations of the middle-graph matrix are examined to determine what survives the subsequent matrix multiplication.

Within graph-based cryptography itself, Ni et al. used special graph families and algebraic graph properties to construct message encryption schemes, whereas Razaq et al. used coset graphs together with symmetric group actions to construct substitution boxes [4,5]. These works illustrate two important roles for graphs in cryptographic design; a graph may serve as the representation through which message symbols are transformed, or it may be used to construct a cryptographic component. More recent proposals extend this direction to uncertainty-weighted graph models. Meenakshi et al. combined intuitionistic fuzzy graphs with domination, vertex strength categorization, and coloring in a multilayer encryption framework, while Annamalai et al. used fuzzy hypergraph and dual-hypergraph structures, fuzzy weights, and additional simulated elements in an encryption–decryption procedure [6,7]. These studies are particularly relevant to the candidate considered here because they place graph weights and structural complexity close to the security mechanism. Our investigation addresses the complementary issue that must also be checked in such constructions: whether the algebraic representation contains invariant or recoverable relations that remain visible to an attacker despite the apparent structural complexity.

This distinction also explains why purely empirical output statistics are insufficient for the present problem. Shannon entropy quantifies uncertainty for a specified random

variable [16], and the avalanche criterion of Webster and Tavares is a useful design criterion for cryptographic components such as substitution boxes [17]. Entropy, correlation, histograms, and avalanche measurements can therefore describe the selected statistical properties of an implementation or output sample, but they do not test the specific linear algebraic key recovery mechanism studied in this paper. For that reason, the security assessment below is based first on an explicit ciphertext-visible relation and the resulting recovery procedure. Empirical measures are treated only as descriptive quantities and not as substitutes for an adversarial security argument.

This paper begins with the middle graph $M(S_{n+1})$ of a star. The proposed fuzzy adjacency scheme chooses some plaintext-dependent weightings for the leaves and edge vertices, performs normalization of the weights, and enciphers the message by multiplying from the right by an invertible matrix. The graph is quite complex as far as the edge vertices inducing a clique, but the rows corresponding to the leaves have a single support even before multiplication, and the central row belongs to their linear span. The latter property survives right multiplication by an invertible matrix. We turn this observation into a precise partial key recovery theorem.

This negative outcome modifies the goal of our investigation. Rather than trying to fix the inherently flawed linear matrix cipher using an additional matrix, we divide the tasks into two different and legitimate ones:

- (i) A *canonical graph encoding* designed to provide an exact and reversible representation together with algebraic validation;
- (ii) An *authenticated encryption layer*, which is responsible for ensuring confidentiality, integrity, and protection against active attacks.

The improved graph layer operates on $\mathbb{F}_{65521}$ and relies on a constant vertex ordering while also being capable of processing any binary sequence, as opposed to only alphabetical sequences. There is no secret matrix involved, and there is no privacy statement either. The codeword used here provides all the necessary information regarding the weighted middle graph. When protected transport is required, the framed codeword is passed to a standardized authenticated encryption with associated data (AEAD) algorithm, such as GCM or Ascon-AEAD128 [18–20].

The principal contributions are deliberately centered on the structural analysis rather than on a new encryption primitive:

- (i) We completely specify the authors' earlier candidate matrix cipher and formalize a ciphertext-only leakage experiment for it.
- (ii) We prove an exact row relation that recovers every y_i and n complete key row from one ciphertext, and we state explicitly what a single ciphertext does not reveal.
- (iii) We provide an executable recovery procedure, a worked numerical attack, an $O(n^3)$ cost bound, and measured attack times.
- (iv) We quantify finite-precision robustness through a singular-value perturbation bound, an exact sufficient condition for recovering the integer y_i values via rounding, and controlled rounded-ciphertext experiments.
- (v) We recast the finite-field construction as a public canonical encoding. Its compact residue vector is primary, while the middle-graph matrix is an optional expanded view for graph-domain processing. Correctness, validation, and random-error detection are representation properties and not security claims.
- (vi) We benchmark this representation against a simpler direct 16-bit encoding and benchmark direct AES-GCM against encoding-plus-AES-GCM, reporting both medians and interquartile ranges.

The main scientific result is therefore the structural cryptanalysis and its robustness analysis. The public middle-graph encoding is retained as a reproducible representation layer, whose use is justified only when a downstream graph application benefits from an adjacency matrix view.

2. Graph Model and Notation

2.1. The Canonically Indexed Middle Graph

Let S_{n+1} be the star with a central vertex c , leaves $\ell_1, \dots, \ell_n$, and edges $c\ell_i$. Its middle graph $M(S_{n+1})$ is obtained by subdividing every star edge and joining two subdivision vertices whenever their corresponding star edges share an endpoint [21]. We denote by e_i the vertex corresponding to the edge $c\ell_i$. Then, we have

$$V(M(S_{n+1})) = \{c\} \cup \{\ell_i : 1 \leq i \leq n\} \cup \{e_i : 1 \leq i \leq n\},$$

and

$$E(M(S_{n+1})) = \{ce_i, \ell_i e_i : 1 \leq i \leq n\} \cup \{e_i e_j : 1 \leq i < j \leq n\}. \quad (1)$$

Thus, $\deg(\ell_i) = 1$, $\deg(c) = n$, and $\deg(e_i) = n + 1$. The edge vertices induce the clique K_n . No embedding, clockwise orientation, or selected geometric cycle is part of the abstract graph.

Definition 1 (Canonical order). *For every n , the vertex order is fixed to*

$$\mathcal{V}_n = (c, \ell_1, \dots, \ell_n, e_1, \dots, e_n).$$

Clique edges are processed in lexicographic order (e_i, e_j) with $i < j$.

Definition 1 removes the rotational and reflection ambiguity of a drawn star. Figure 1 illustrates the labels for $n = 4$; the labels, rather than the drawing, determine all computations. Throughout the paper, a matrix row subscript names the vertex indexing that row. In particular, $B_{e_i,*}$ denotes the row of the secret matrix indexed by the edge vertex e_i , whereas $B_{\ell_i,*}$ denotes the leaf-indexed row; these rows are distinct.

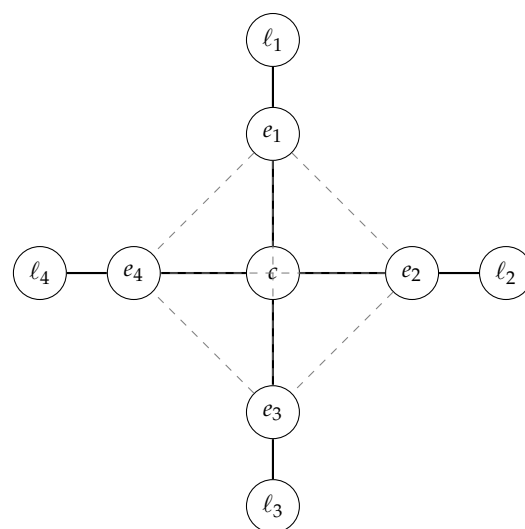


Figure 1. Canonical labels for $M(S_5)$. Solid edges connect the central vertex or a leaf to its associated edge vertex; dashed edges form the clique on e_1, e_2, e_3, e_4 . The planar placement has no algorithmic meaning.

2.2. Weighted Adjacency Matrices

Let R be a commutative ring, and let $w : E(G) \rightarrow R$. Relative to an ordered vertex list $(v_1, \dots, v_d)$, the weighted adjacency matrix $A_w \in R^{d \times d}$ is defined by

$$(A_w)_{ij} = \begin{cases} w(v_i v_j), & v_i v_j \in E(G), \\ 0, & v_i v_j \notin E(G). \end{cases}$$

For an undirected graph, A_w is symmetric. A fuzzy graph is obtained when the vertex and edge memberships satisfy the usual constraints in $[0, 1]$ [22–24]. The structural attack in Section 3 needs only the support pattern of certain rows; it is unaffected by the terminology used for the normalized weights.

2.3. Threat Model, Leakage Experiments, and the Encoding–Encryption Distinction

An *encoding* is a public injective map that represents data in another format. Anyone can decode it. Encryption, by contrast, is keyed, and it is evaluated against an adversary who knows the algorithm and may obtain ciphertexts or plaintext–ciphertext pairs according to the stated model [1,3]. In authenticated encryption, successful decryption additionally certifies the ciphertext and associated data [25–27].

For the attacked candidate, we use two explicit experiments. In the *ciphertext-only leakage experiment* CO-Leak_n, the challenger generates a valid candidate instance and an invertible key B , returns only (C, m) , and the adversary wins if it outputs the complete vector $(y_1, \dots, y_n)$ and the key-row block $(B_{e_1,*}, \dots, B_{e_n,*})$. Theorem 1 gives a deterministic adversary that wins this experiment with a probability of one in exact arithmetic for every valid instance.

In the *known-plaintext key reuse experiment* KP-Reuse_n, the same key B is used for known plaintext–ciphertext pairs $(A^{(j)}, C^{(j)})$. Stacking the equations $A^{(j)} B = C^{(j)}$ gives a unique B exactly when the stacked coefficient matrix has a full column rank $2n + 1$. Two pairs suffice whenever their one-dimensional nullspaces are different; more pairs can be added until the rank condition is met. This rank condition, rather than a vague claim that any two different plaintexts suffice, is the exact recovery criterion.

We use the following terminology throughout:

- (i) The *candidate cipher* is the authors' earlier normalized adjacency construction followed by $C = AB$;
- (ii) The *canonical public encoder* is the exact map EncG_n defined in Section 4;
- (iii) Φ_n is an optional expanded middle-graph matrix reconstructed from the compact codeword;
- (iv) *Protected transport*, when needed, is an independent AEAD algorithm applied to the framed compact codeword.

This separation prevents graph symmetry, matrix size, redundancy, or validation equations from being mistaken for confidentiality or adversarial integrity.

3. Structural Cryptanalysis of the Candidate Matrix Cipher

3.1. Provenance and Complete Specification of the Attacked Candidate

The attacked scheme is the authors' own candidate construction from an earlier version of this manuscript; it is not presented as a published cipher. We state it completely here so that the cryptanalytic target is reproducible.

For a block of $n \geq 3$ alphabetic symbols, letters are mapped to $k_i \in \{1, \dots, 26\}$ and shifted publicly by the block length, where $x_i = (k_i + n) \bmod 26$, with a residue of 0 represented by 26. We set $y_0 = 26$ and choose

$$y_i = \min\{t \in \mathbb{Z}_{>0} : t > y_{i-1}, \gcd(t, x_i) = 1\}, \quad 1 \leq i \leq n, \quad (2)$$

Then, $z_i = x_i^{-1} \pmod{y_i}$.

Relative to the canonical vertex order, we assign y_i to ℓ_i , z_i to e_i , and n to c . The leaf edge $\ell_i e_i$ receives weight y_i , and the central edge ce_i receives weight n . The earlier drawing-based clique rule is fixed here canonically. Let $H_n = (e_1 e_2, e_2 e_3, \dots, e_{n-1} e_n, e_n e_1)$ be the selected clique cycle, assign weight z_i to $e_i e_{i+1}$ (indices modulo n), and assign every remaining clique edge $e_i e_j$ the weight $\max\{z_i, z_j\}$. This canonicalization only removes an ambiguity in the earlier description; the attack below uses none of the clique weights.

Let

$$m = \max\{n, y_1, \dots, y_n, z_1, \dots, z_n\}$$

and divide every edge weight by m to obtain $A \in \mathbb{Q}^{(2n+1) \times (2n+1)}$. The secret matrix $B \in \mathbb{Z}^{(2n+1) \times (2n+1)}$ is generated by sampling entries independently from $\{0, \dots, 99\}$ and rejecting until $\text{rank}_{\mathbb{Q}}(B) = 2n + 1$. Encryption is

$$C = AB. \quad (3)$$

The public output is (C, m) , and B is secret. The formal attack treats C as an exact rational matrix. Section 3.3 separately quantifies what happens when transmitted entries are rounded to a fixed number of decimal places. Thus, the cryptanalytic result does not depend on an unspecified binary serialization.

The following theorem shows that right multiplication does not hide the decisive row relation.

Theorem 1 (Ciphertext-visible row relation). *Order the vertices canonically. For every valid instance of the candidate cipher, we have*

$$A_{c,*} = \sum_{i=1}^n \frac{n}{y_i} A_{\ell_i,*} \quad (4)$$

and

$$C_{c,*} = \sum_{i=1}^n \frac{n}{y_i} C_{\ell_i,*}. \quad (5)$$

Moreover, the rows $C_{\ell_1,*}, \dots, C_{\ell_n,*}$ are linearly independent. Hence, the coefficients in Equation (5) are unique and can be recovered from a single ciphertext via linear algebra.

Proof. The row $A_{\ell_i,*}$ has exactly one nonzero entry: its coordinate at e_i is y_i/m . The row $A_{c,*}$ is nonzero at the same n coordinates, and its coordinate at every e_i is n/m . Therefore, we have

$$\frac{n}{y_i} (A_{\ell_i,*})_{e_i} = \frac{n}{m},$$

while all cross-coordinates vanish, proving Equation (4). Right multiplication by B gives Equation (5).

Furthermore, the only nonzero entry of $A_{\ell_i,*}$ lies in column e_i . Therefore, the matrix product selects the e_i -indexed row of B (not the ℓ_i -indexed row and not the central row):

$$C_{\ell_i,*} = A_{\ell_i,*} B = \frac{y_i}{m} B_{e_i,*}. \quad (6)$$

Thus, Equation (6), Corollary 1, and the recovery algorithm all refer to the same edge-vertex row $B_{e_i,*}$. Distinct rows of an invertible matrix are linearly independent. Nonzero scalar multiplication preserves independence, and thus the n ciphertext rows in Equation (6) are linearly independent. The representation of $C_{c,*}$ in their span is consequently unique. $\square$

Corollary 1 (Partial key recovery). *Let α_i be the unique solution of*

$$C_{c,*} = \sum_{i=1}^n \alpha_i C_{\ell_i,*}.$$

Then, a ciphertext-only adversary obtains

$$y_i = \frac{n}{\alpha_i}, \quad B_{e_i,*} = \frac{m}{y_i} C_{\ell_i,*}, \quad 1 \leq i \leq n. \quad (7)$$

Thus, one ciphertext discloses every y_i and n complete rows of the secret matrix B .

The leakage in Corollary 1 is substantial but should not be overstated. A single ciphertext determines the y_i values and the n rows $B_{e_i,*}$, but it does *not* by itself determine the message-dependent values x_i or z_i . Those quantities influence other rows of A and remain mixed by the unknown part of B . Message recovery or completion of the remaining key therefore requires additional information, such as known plaintexts under key reuse.

The row relation also proves that both A and C are singular. This gives an encoding-independent distinguisher from a generic full-rank matrix and shows why sample entropy or visual complexity cannot remove the algebraic leakage.

3.2. Executable Ciphertext-Only Recovery and Computational Cost

Let $R \in \mathbb{R}^{n \times (2n+1)}$ contain the leaf ciphertext rows $C_{\ell_i,*}$, and let $c = C_{c,*}^T$. The recovery algorithm is as follows:

- (1) Solve the full-column-rank system $R^T \alpha = c$ for $\alpha = (\alpha_1, \dots, \alpha_n)^T$;
- (2) Recover $y_i = n / \alpha_i$;
- (3) Recover $B_{e_i,*} = (m / y_i) C_{\ell_i,*}$.

With $d = 2n + 1$, QR factorization or Gaussian elimination costs $O(dn^2) = O(n^3)$ arithmetic operations and stores $O(nd) = O(n^2)$ numbers. The subsequent parameter and key-row recovery is $O(n^2)$.

Example 1 (Exact recovery from a small ciphertext). *Take $n = 3$, $(y_1, y_2, y_3) = (27, 29, 31)$, $m = 31$, and the invertible lower-triangular key*

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 2 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad \det B = 1.$$

Only four ciphertext rows are needed:

$$\begin{aligned}
C_{\ell_1,*} &= \frac{27}{31}(2, 1, 0, 0, 1, 0, 0), \\
C_{\ell_2,*} &= \frac{29}{31}(0, 2, 1, 0, 0, 1, 0), \\
C_{\ell_3,*} &= (1, 0, 2, 1, 0, 0, 1), \\
C_{c,*} &= \frac{3}{31}(3, 3, 3, 1, 1, 1, 1).
\end{aligned}$$

Solving $R^T \alpha = c$ gives

$$\alpha = \left(\frac{3}{27}, \frac{3}{29}, \frac{3}{31} \right)^T.$$

Hence, $y_i = 3/\alpha_i$ gives $(y_1, y_2, y_3) = (27, 29, 31)$, and scaling the three leaf rows by $31/y_i$ recovers exactly the three edge-vertex rows $B_{e_1,*}, B_{e_2,*}, B_{e_3,*}$. No search over the nominal matrix-key space is involved.

A NumPy implementation of this recovery was timed on Linux 6.18, Python 3.13.5, NumPy 2.3.5, and an AMD EPYC 9V74 processor. Table 1 reports the median and interquartile range (IQR) over 11 batches.

Table 1. Ciphertext-only coefficient recovery time for one ciphertext. Recovery times are median (IQR) microseconds over 11 timing batches for one fixed randomly generated key at each n . The reported $\kappa_2(R)$ is the condition number of that same timed instance; it is not an ensemble median.

n	Recovery Time (μ s)	$\kappa_2(R)$ for Timed Instance
8	18.35 (1.21)	11.27
16	40.55 (0.58)	23.59
32	135.98 (9.23)	36.40
64	583.77 (59.25)	62.52

These timings are implementation-specific, but they confirm that the attack is a small linear-algebra computation rather than a brute force search.

3.3. Finite-Precision Robustness

The exact theorem does not imply that every arbitrarily coarse serialization preserves recoverability. The relevant quantity is the conditioning of the exposed leaf-row system. Let the exact relation be $R^T \alpha = c$, and suppose that the observed values are $\tilde{R} = R + E$ and $\tilde{c} = c + e$. Let $\hat{\alpha}$ be the least squares solution of $\tilde{R}^T \alpha \approx \tilde{c}$.

Proposition 1 (Perturbation bound). *If $\|E\|_2 < \sigma_{\min}(R)$, then $\tilde{R}$ retains full row rank, and*

$$\|\hat{\alpha} - \alpha\|_2 \leq \frac{\|e\|_2 + \|E\|_2 \|\alpha\|_2}{\sigma_{\min}(R) - \|E\|_2}. \quad (8)$$

Moreover, if the component error satisfies

$$|\hat{\alpha}_i - \alpha_i| < \frac{n}{y_i(2y_i + 1)}, \quad (9)$$

then rounding $n/\hat{\alpha}_i$ to the nearest integer recovers y_i exactly.

Proof. Since $\tilde{c} = \tilde{R}^T \alpha + (e - E^T \alpha)$, and $\tilde{R}^T$ has full column rank, the least squares solution satisfies $\hat{\alpha} - \alpha = (\tilde{R}^T)^\dagger (e - E^T \alpha)$. Weyl's inequality gives $\sigma_{\min}(\tilde{R}) \geq \sigma_{\min}(R) - \|E\|_2$, which yields Equation (8). For the integer recovery, $\alpha_i = n/y_i$, and

$$\left| \frac{n}{\hat{\alpha}_i} - y_i \right| = \frac{n|\hat{\alpha}_i - \alpha_i|}{|\alpha_i \hat{\alpha}_i|}.$$

Let $\delta_i = |\hat{\alpha}_i - \alpha_i|$, and assume Equation (9). Since $\alpha_i = n/y_i > 0$, we have

$$\hat{\alpha}_i \geq \alpha_i - \delta_i > \frac{n}{y_i} - \frac{n}{y_i(2y_i + 1)} = \frac{2n}{2y_i + 1} > 0.$$

Consequently, we have

$$|\alpha_i \hat{\alpha}_i| = \alpha_i |\hat{\alpha}_i| > \frac{n}{y_i} \frac{2n}{2y_i + 1} = \frac{2n^2}{y_i(2y_i + 1)}.$$

Substituting these two strict bounds into the preceding identity gives

$$\left| \frac{n}{\hat{\alpha}_i} - y_i \right| < \frac{n[n/(y_i(2y_i + 1))]}{2n^2/[y_i(2y_i + 1)]} = \frac{1}{2}.$$

Thus, nearest-integer rounding returns y_i . Equation (9) is a uniform two-sided sufficient condition; it is not claimed to be necessary or sharp for every perturbation direction. $\square$

Here, $R = DB_E$, where $D = \text{diag}(y_1/m, \dots, y_n/m)$ and B_E is the $n \times (2n + 1)$ block consisting of the edge-vertex rows of B . Consequently, we have

$$\kappa_2(R) \leq \frac{y_{\max}}{y_{\min}} \kappa_2(B_E),$$

and thus numerical robustness is governed by the exposed key-row block and the scale ratio, not by the visual complexity of the graph.

To quantify the effect of rounding, we generated 200 independent full-rank integer keys for each $n \in \{8, 16, 32, 64\}$, used $y_i = 26 + i$, formed the exact rows, rounded every observed ciphertext entry to p decimal places, solved by least squares, and tested exact integer recovery of the entire y vector. The seed and code are supplied in `scripts/reviewer_experiments.py` and the Recovery of the complete y vector from rounded ciphertext rows over 200 independently generated keys can be seen in Table 2.

Table 2. Recovery of the complete y vector from rounded ciphertext rows over 200 independently generated keys for each n . Here, median $\kappa_2(R)$ is the median across those 200 keys; unlike Table 1, it is not the condition number of a single timed instance.

n	Median $\kappa_2(R)$	$p = 0$	$p = 1$	$p = 2$
8	11.82	10.0%	100.0%	100.0%
16	19.56	2.5%	100.0%	100.0%
32	33.26	0.0%	100.0%	100.0%
64	59.92	0.0%	97.5%	100.0%

The experiment does not claim universal recovery under arbitrary quantization; instead, it identifies a measurable precision–conditioning tradeoff and supports the analytic bound. In these instances, two decimal places were sufficient in every trial, while integer-only rounding was generally too coarse.

3.4. Known-Plaintext Reuse and Fresh-Key Overhead

If B is reused for messages of the same length, then Corollary 1 exposes the same n key rows repeatedly. For a known plaintext, the public preprocessing gives x_i ; after recovering y_i , one obtains $z_i = x_i^{-1} \pmod{y_i}$ and reconstructs the full A . For t known pairs, we define

$$\mathcal{A}_t = \begin{pmatrix} A^{(1)} \\ \vdots \\ A^{(t)} \end{pmatrix}, \quad \mathcal{C}_t = \begin{pmatrix} C^{(1)} \\ \vdots \\ C^{(t)} \end{pmatrix}.$$

Then, B is uniquely determined by $\mathcal{A}_t B = \mathcal{C}_t$ if and only if $\text{rank}(\mathcal{A}_t) = 2n + 1$. Because each individual $A^{(j)}$ is singular, one pair cannot determine all rows. Two pairs suffice whenever their nullspaces are distinct; otherwise, additional pairs are required. This statement cleanly separates ciphertext-only partial leakage from known-plaintext completion of the break.

Generating an independent B for every message prevents cross-message accumulation but does not solve key distribution. The parties would have to derive or confidentially deliver $(2n + 1)^2$ matrix entries for only n symbols. Under the earlier $\{0, \dots, 99\}$ integer-key rule, a fixed-width representation needs at least $7(2n + 1)^2$ bits merely to store the sampled entries; a binary64 implementation uses $64(2n + 1)^2$ bits. For $n = 30$, binary64 therefore uses 238,144 key bits for 30 input symbols. Protecting or deriving a fresh secret object of that size already requires an established secure key management mechanism, and thus the matrix cannot remove the need for conventional cryptography. By comparison, standardized AEAD systems operate with compact 128- or 256-bit keys and well-defined nonce and key management requirements.

3.5. Why Empirical Output Statistics Do Not Repair the Structural Attack

The structural recovery above is independent of the following sample statistics:

- (i) Plaintext entropy and scaled ciphertext entropy are computed on different alphabets and different sample lengths; their ratio has no operational security interpretation.
- (ii) Repeating a short plaintext until it matches a flattened matrix creates an artificial paired sample. A near-zero Pearson correlation of that sample does not imply indistinguishability.
- (iii) A bitwise avalanche calculation is undefined until real entries have a fixed, canonical binary serialization. Values between 21% and 26% would, in any case, be substantially below the usual one-half design target.
- (iv) A broad histogram cannot rule out the exact linear relation in Theorem 1.
- (v) Counting matrices with entries in $\{0, \dots, 99\}$ estimates a search space and not the work factor of the algebraic recovery in Equation (7).

Accordingly, the remainder of the paper makes no security inference from entropy, correlation, histograms, or visual output complexity.

4. Canonical Public Middle-Graph Encoding and Optional Matrix Expansion

4.1. Parameters and Byte Map

Fix

$$q = 65,521,$$

the largest prime below 2^{16} , and use $\mathbb{F}_q$ as a common exact storage domain for the optional graph matrix. The field size is a representation choice and not a security parameter. For the present file format, we choose $1 \leq n \leq 256$, and let $r_1 < \dots < r_n$ be the first n primes

strictly greater than 257. Here, $r_{256} = 2063$, and thus every tag and pairwise check fits in one unsigned 16-bit field element. The value 256 is therefore an engineering bound aligned with byte-oriented blocks and not a graph-theoretic limit. Larger blocks could use a larger field or another public tag schedule. If no graph consumer requires coordinate tags at all, then length-prefixed raw bytes or direct AEAD are simpler alternatives and are explicitly benchmarked below.

For a byte block

$$\mathbf{b} = (b_1, \dots, b_n) \in \{0, \dots, 255\}^n,$$

set

$$x_i = b_i + 1 \in \{1, \dots, 256\}, \quad z_i = x_i^{-1} \pmod{r_i}, \quad (10)$$

where z_i is represented by its integer residue in $\{1, \dots, r_i - 1\}$ and then embedded into $\mathbb{F}_q$. Because $r_i > 257$, every x_i is invertible modulo r_i . Coordinate-wise modular inversion is a public bijective relabeling of the byte alphabet. We do not claim this elementary bijection as a cryptographic primitive; its role is to create canonical residues from which the middle-graph representation is reconstructed.

4.2. Primary Compact Codeword and Optional Matrix Expansion

The primary serialized block representation is

$$\text{EncG}_n(\mathbf{b}) = (z_1, \dots, z_n). \quad (11)$$

The topology, vertex order, tag schedule, and clique rule are public, and thus this length- n codeword uniquely determines the optional expanded matrix. The matrix should be materialized only when a graph- or matrix-domain consumer needs an adjacency representation.

Definition 2 (Optional canonical matrix expansion). *The matrix $\Phi_n(\mathbf{b}) = [a_{uv}] \in \mathbb{F}_q^{(2n+1) \times (2n+1)}$ is indexed by $\mathcal{V}_n$ and is symmetric with zero diagonal. Its nonzero edge entries are*

$$a_{c,e_i} = z_i, \quad 1 \leq i \leq n, \quad (12)$$

$$a_{\ell_i,e_i} = r_i, \quad 1 \leq i \leq n, \quad (13)$$

$$a_{e_i,e_j} = z_i + z_j, \quad 1 \leq i < j \leq n. \quad (14)$$

All non-edge entries are zero.

The clique values in Equation (14) are redundant; they can be recomputed from the central-edge values. They provide algebraic consistency checks when the dense matrix is used for analysis or storage. They do not provide authentication against an adversary who can modify several coordinates coherently.

Proposition 2 (Canonical block-encoding procedure). *Given $\mathbf{b} = (b_1, \dots, b_n)$ with $1 \leq n \leq 256$, process the coordinates in increasing index order. For every i , set $x_i \leftarrow b_i + 1$ and $z_i \leftarrow x_i^{-1} \pmod{r_i}$. Return the compact codeword $\mathbf{z} = (z_1, \dots, z_n)$.*

4.3. Decoder and Structural Validation

Given a compact coordinate z_i , compute

$$\hat{x}_i = z_i^{-1} \pmod{r_i}, \quad \hat{b}_i = \hat{x}_i - 1. \quad (15)$$

The coordinate is accepted only if $1 \leq z_i < r_i$ and $1 \leq \hat{x}_i \leq 256$.

For a dense matrix, the validator additionally checks its order, symmetry, zero diagonal, zero non-edge entries, leaf tags (Equation (13)), and all clique identities (Equation (14)). The decoder rejects before returning bytes when any check fails.

Proposition 3 (Compact block-decoding procedure). *Given $\mathbf{z} = (z_1, \dots, z_n)$, process the coordinates in increasing index order. Reject them if $z_i \notin \{1, \dots, r_i - 1\}$. Otherwise, compute $x_i \leftarrow z_i^{-1} \pmod{r_i}$, and reject them if $x_i \notin \{1, \dots, 256\}$. Set $b_i \leftarrow x_i - 1$. If every coordinate is accepted, then return $\mathbf{b} = (b_1, \dots, b_n)$.*

Proposition 4 (Exact correctness of the public encoding). *For every $1 \leq n \leq 256$ and every $\mathbf{b} \in \{0, \dots, 255\}^n$, we have*

$$\text{DecG}_n(\text{EncG}_n(\mathbf{b})) = \mathbf{b}.$$

Proof. For each coordinate, $x_i = b_i + 1$ lies in $\{1, \dots, 256\}$ and is nonzero modulo the prime $r_i > 257$. Hence, $z_i = x_i^{-1} \pmod{r_i}$ exists. Applying the decoder yields

$$z_i^{-1} \equiv (x_i^{-1})^{-1} \equiv x_i \pmod{r_i}.$$

Both x_i and the decoder output belong to $\{1, \dots, 256\}$, a set of representatives strictly smaller than r_i , and thus congruence implies equality. Therefore, $\hat{b}_i = x_i - 1 = b_i$ for every i . $\square$

Corollary 2 (Injectivity). *The maps EncG_n and Φ_n are injective on $\{0, \dots, 255\}^n$.*

Proof. A left inverse exists due to Proposition 4; hence, EncG_n is injective. The coordinates z_i occur explicitly at (c, e_i) in Φ_n , and thus equality of matrices implies equality of compact codewords. $\square$

The proposition and corollary are elementary representation guarantees; they establish unambiguous round-trip decoding and not cryptographic hardness.

Remark 1 (What the validator guarantees). *The validation equations detect malformed or accidentally corrupted representations that do not satisfy the public code constraints. They do not prevent a deliberate replacement of one valid codeword by another. Adversarial integrity must be supplied by the authenticated encryption layer in Section 5.*

Proposition 5 (Random-error detection of redundant clique checks). *Fix the residue coordinates $z_1, \dots, z_n$. Suppose that $s \geq 1$ stored clique check entries are independently replaced by uniformly random elements of $\mathbb{F}_q$, while the corresponding residues are unchanged. Then, the probability that all s corrupted checks accidentally satisfy Equation (14) is q^{-s} ; hence, the detection probability is $1 - q^{-s}$. For one random replacement and $q = 65,521$, this is approximately 99.99847%.*

Proof. For each affected edge, there is exactly one field value equal to the required sum $z_i + z_j$. A uniform replacement hits that value with a probability $1/q$, and independence gives q^{-s} for s checks. $\square$

This proposition concerns an explicit non-adversarial random-substitution model only. An adversary who changes residues and recomputes all dependent checks can always construct another valid codeword, and thus AEAD authentication remains necessary against active manipulation.

4.4. Concrete Graph-Domain Use Case and Scope of Utility

A concrete reason to reconstruct Φ_n is interoperability with a downstream graph analysis interface that accepts weighted adjacency matrices. For example, after an authenticated packet is opened, a graph monitoring or provenance pipeline can reconstruct Φ_n and compute weighted degrees, Laplacian features, or spectral moments using standard graph software. For the SNOW example in Section 7, the weighted-degree vector of the canonical matrix is

$$(568, 263, 269, 271, 277, 1404, 1215, 1337, 1100).$$

The same values are, of course, derivable directly from the residue vector; the graph does not create new information or a faster algorithm. Its benefit in this scenario is a canonical adjacency matrix interface with explicit vertex roles and check relations. When no downstream computation consumes graph structure, the manuscript recommends keeping only the compact codeword or omitting this public re-encoding entirely and applying AEAD directly to the original bytes.

5. Lossless Framing and Optional Authenticated Transport

5.1. Arbitrary Binary Input

The alphabet-only preprocessing of the candidate cipher is lossy; spaces, punctuation, capitalization, digits, and non-English characters cannot be recovered. The revised encoder operates on bytes. Text is first encoded in a declared character encoding such as UTF-8; images, archives, and other binary files are already byte strings.

Let a message contain L bytes. A 16-byte public header contains the following:

- (i) The four-byte format identifier MGE1;
- (ii) A two-byte block length n ;
- (iii) Two reserved flag bytes;
- (iv) An unsigned eight-byte big-endian value L .

The message is zero-padded to kn bytes, where $k = \lceil L/n \rceil$. The stored length removes the padding exactly. Each block is mapped to n residues, and each residue is serialized as an unsigned two-byte big-endian integer. The compact encoded payload therefore contains $2kn$ bytes.

5.2. Optional AEAD Security Boundary

For protected transport, let (Seal, Open) be a nonce-based AEAD scheme. The format header is associated data, and the compact graph payload is the AEAD plaintext. The graph transform is public and reversible. When secure transport is required, the AEAD primitive alone supplies confidentiality and adversarial integrity can be seen in Figure 2.

$$(C, T) \leftarrow \text{Seal}_K(N, \text{Enc}_{G_n}^{\text{msg}}(M), H), \quad (16)$$

$$\text{Enc}_{G_n}^{\text{msg}}(M) \leftarrow \text{Open}_K(N, C, T, H). \quad (17)$$

Only after authentication succeeds is $\text{Dec}_{G_n}^{\text{msg}}$ applied. A unique nonce N is required for each encryption under the same key when mandated by the selected AEAD specification. GCM is standardized in NIST SP 800-38D, while Ascon-AEAD128 is standardized for constrained devices in NIST SP 800-232 [18,19]. The AEAD interface and the role of associated data are described in RFC 5116 [20].

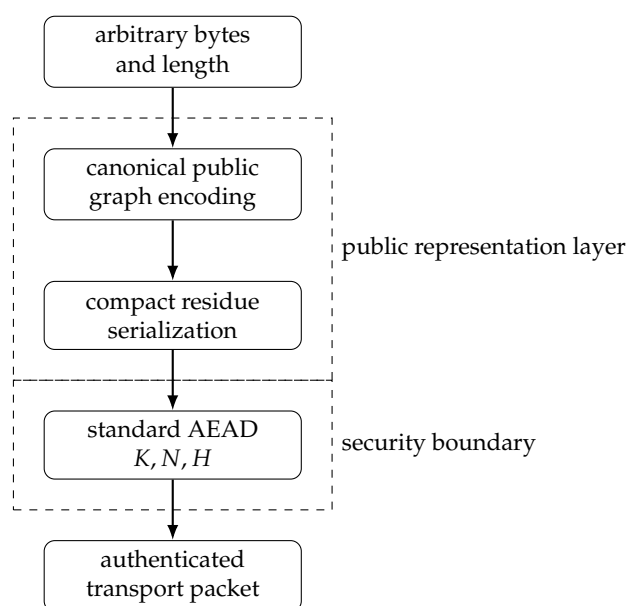


Figure 2. Separation of responsibilities. The graph transform is public and reversible. When secure transport is required, the AEAD primitive alone supplies confidentiality and adversarial integrity.

5.3. Security Inheritance as a Boundary Check

Proposition 6 (Security-preserving public preprocessing). *Let $\text{EncG}_n^{\text{msg}}$ be the deterministic injective message framing and compact encoding above, and suppose that equal-length messages produce equal-length encodings. If AEAD is secure for nonce-respecting confidentiality and ciphertext integrity, then the composition in Equations (16) and (17) has the same properties up to the length and public header information intentionally revealed by the format.*

Proof. Assume that an adversary distinguishes encryptions of two equal-length original messages in the composed system. A reduction forwards their deterministic encodings to the AEAD challenge oracle, uses the same public header as associated data, and returns the resulting challenge unchanged. Because the message encoding is public and deterministic, the simulated view is identical to the real composed view. The distinguishing advantage is therefore the same as the advantage against AEAD confidentiality.

For integrity, the decoder is invoked only after Open accepts. Any new packet accepted by the composed receiver therefore contains an AEAD ciphertext–tag pair accepted for the corresponding nonce and header. A successful composed forgery yields an AEAD forgery. Injectivity ensures that an accepted encoded payload has at most one decoded byte string. No additional security term is attributed to the graph layer. $\square$

The proposition is intentionally conservative and is not claimed as a new cryptographic primitive or a graph-specific security property. It states only that the public representation does not replace AEAD and does not amplify the underlying key strength. The effective confidentiality and integrity are exactly those of the selected AEAD algorithm, implementation, nonce discipline, and key management procedure.

6. Complexity and Representation Expansion

6.1. Time and Space Complexity

The tag primes are fixed and may be tabulated once. For fixed 16-bit moduli, each coordinate inversion has a bounded word cost. Thus, compact block encoding and decoding use $O(n)$ word operations and $O(n)$ space. Under a bit complexity model, an extended Euclidean inversion modulo r_i takes $O(\log r_i)$ arithmetic steps, giving $O(n \log q)$ for a block [28].

Materializing the full matrix requires $\binom{n}{2}$ clique weights and $(2n + 1)^2$ stored entries, hence having $O(n^2)$ time and space. Dense validation is also $O(n^2)$. Compact AEAD processing is linear in the encoded byte length for standard streaming implementations; its exact constant depends on the selected algorithm and platform.

For the public encoder, no dense matrix multiplication or matrix inversion is part of the compact path. Compact encoding is therefore linear in the number of coordinates; the optional dense expansion is quadratic because it materializes all matrix entries and clique checks.

6.2. Expansion Formulas

Let $d = 2n + 1$. Table 3 distinguishes four representations. The original estimate assumes binary64 storage for every real matrix entry. The exact dense format uses two bytes per $\mathbb{F}_q$ entry. In the sparse format, the fixed topology means indices need not be transmitted; only the $2n + \binom{n}{2}$ edge weights are serialized. The compact codeword transmits only $z_1, \dots, z_n$.

Table 3. Storage per full n -byte block before general protocol framing.

Representation	Stored Content	Bytes
Candidate dense real matrix	$(2n + 1)^2$ binary64 entries	$8(2n + 1)^2$
Exact dense field matrix	$(2n + 1)^2$ unsigned 16-bit entries	$2(2n + 1)^2$
Fixed-topology sparse edges	$2n + \binom{n}{2}$ unsigned 16-bit weights	$n(n + 3)$
Compact graph codeword	n unsigned 16-bit residues	$2n$

For a one-block AEAD packet with a 16-byte header, a 12-byte nonce, and a 16-byte tag, the total compact size is

$$2n + 44 \text{ bytes.} \quad (18)$$

For multiple blocks, the 44-byte fixed cost is amortized over the message. The ratios are relative to an n -byte input block can be seen in Table 4.

Table 4. Representative expansion values. Ratios are relative to an n -byte input block.

n	Candidate	Ratio	Exact Dense	Ratio	Compact	AEAD Packet	Ratio
16	8712	544.50	2178	136.13	32	76	4.75
30	29,768	992.27	7442	248.07	60	104	3.47
32	33,800	1056.25	8450	264.06	64	108	3.38
64	133,128	2080.13	33,282	520.03	128	172	2.69

The quadratic formats become less efficient as n grows, whereas the compact graph codeword has an intrinsic factor of two. This accounting is essential; a large matrix can create a large nominal key space and high sample entropy simply because it contains far more symbols than the input. The expansion grows linearly in n for dense representations but remains constant for compact serialization, apart from the fixed packet overhead. The vertical axis is logarithmic can be seen in Figure 3.

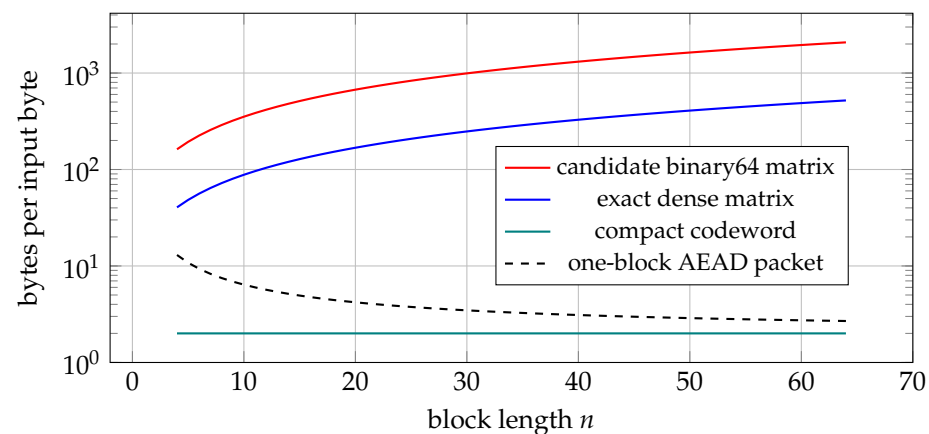


Figure 3. Expansion grows linearly in n for dense representations but remains constant for compact serialization, apart from the fixed packet overhead. The vertical axis is logarithmic.

7. Numerical Example and Reproducible Evaluation

7.1. Exact Encoding of SNOW

Consider the four-byte ASCII string SNOW. Its bytes and shifted nonzero representatives are

$$\mathbf{b} = (83, 78, 79, 87), \quad \mathbf{x} = (84, 79, 80, 88).$$

The first four tag primes are

$$(r_1, r_2, r_3, r_4) = (263, 269, 271, 277).$$

Using (10),

$$\begin{aligned} \mathbf{z} &= (84^{-1} \bmod 263, 79^{-1} \bmod 269, 80^{-1} \bmod 271, 88^{-1} \bmod 277) \\ &= (191, 126, 166, 85). \end{aligned}$$

For example, $84 \cdot 191 = 16,044 \equiv 1 \pmod{263}$. The clique checks are

$$\begin{pmatrix} 0 & 317 & 357 & 276 \\ 317 & 0 & 292 & 211 \\ 357 & 292 & 0 & 251 \\ 276 & 211 & 251 & 0 \end{pmatrix},$$

whose (i, j) entry is $z_i + z_j$. Relative to the order $(c, \ell_1, \dots, \ell_4, e_1, \dots, e_4)$, the complete matrix is compactly displayed in block form as follows:

$$\Phi_4(\text{SNOW}) = \left(\begin{array}{c|cc} 0 & 0_{1 \times 4} & \mathbf{z} \\ \hline 0_{4 \times 1} & 0_{4 \times 4} & \text{diag}(263, 269, 271, 277) \\ \hline \mathbf{z}^T & \text{diag}(263, 269, 271, 277) & R(\mathbf{z}) \end{array} \right), \quad (19)$$

where $R(\mathbf{z})$ is the displayed clique check matrix. Decoding gives

$$(191^{-1} \bmod 263, 126^{-1} \bmod 269, 166^{-1} \bmod 271, 85^{-1} \bmod 277) = (84, 79, 80, 88),$$

and subtracting one returns $(83, 78, 79, 87)$ exactly.

7.2. Reproducible Implementation and Baseline Evaluation

The supplementary archive supplied with this revision contains an actual `code/scripts/` directory rather than a manuscript file. The script `reference_encoder.py` implements

the direct 16-bit baseline, compact encoding and decoding, optional dense matrix construction, dense validation, and the graph-domain degree example. The script `reviewer_experiments.py` implements the ciphertext-only attack, the 200-key rounded-ciphertext study, and AES-GCM comparisons. The `README.md` and `requirements.txt` give exact commands and dependencies. Fixed seeds are included; the deterministic study seeds reproduce the ensemble condition numbers and recovery percentages in Table 2. Round-trip correctness is checked before timing.

Measurements were obtained on Linux 6.18 with Python 3.13.5 on an AMD EPYC 9V74 processor. NumPy 2.3.5 was used for the linear algebra attack experiments and cryptography 46.0.4 for AES-GCM. Each reported value is the median over 11 batches; the number in parentheses is the interquartile range (IQR). These timings characterize the supplied reference implementation and not a universal performance claim. The direct baseline simply widens each input byte to unsigned 16-bit big-endian form can be seen in Table 5.

Table 5. Public representation costs in microseconds per block, reported as median (IQR). The direct baseline simply widens each input byte to unsigned 16-bit big-endian form.

<i>n</i>	Direct 16-Bit	Graph Encode	Graph Decode	Dense Expand	Dense Validate
8	0.66 (0.01)	1.62 (0.07)	2.13 (0.05)	4.64 (0.21)	10.39 (0.96)
16	1.17 (0.04)	3.08 (0.15)	4.05 (0.14)	15.33 (0.72)	33.88 (0.95)
32	2.11 (0.12)	6.32 (0.43)	8.29 (0.29)	57.18 (1.13)	126.79 (5.01)
64	3.99 (0.99)	11.90 (0.27)	15.71 (0.41)	218.88 (8.36)	501.43 (13.20)

The simpler baseline was consistently faster because it performed no modular inversions. The graph encoder therefore has no performance advantage as a generic byte representation; its extra cost is justified only when the canonical graph view is actually required by a downstream graph consumer.

To isolate the cost of adding the public representation before a standard secure channel, Table 6 compares AES-256-GCM applied directly to the raw byte block with the complete compact-encode-then-AES-GCM path. The same 16-byte header was authenticated as associated data in both cases.

Table 6. Direct AES-256-GCM versus compact graph encoding followed by AES-256-GCM; microseconds per block, median (IQR).

<i>n</i>	Direct AES-GCM	Encode + AES-GCM
8	0.60 (0.01)	3.18 (0.11)
16	0.59 (0.01)	5.13 (0.24)
32	0.61 (0.03)	9.39 (0.40)
64	0.60 (0.05)	17.17 (0.18)

The result is intentionally not presented as a performance win; for ordinary secure transport, direct AEAD is both simpler and faster. The graph layer is a public representation layer for applications that need the middle-graph object; it is not a cryptographic accelerator.

All reported values can be regenerated with

```
python3 scripts/reference_encoder.py -benchmark
python3 scripts/reviewer_experiments.py
```

8. Discussion

8.1. Scientific Contribution and Interpretation

The strongest result of this study is the structural cryptanalysis of a fully specified candidate cipher. The attack exposes a deterministic linear relation that survives secret right

multiplication, gives an exact ciphertext-only leakage algorithm, and remains practically relevant under controlled finite precision. This is a negative result; it demonstrates that graph complexity, normalization, key space counts, entropy, and avalanche-style statistics cannot substitute for an adversarial analysis of the actual algebraic transformation.

The finite-field construction has a deliberately narrower role. The byte-to-residue map is a public coordinatewise bijection, and the compact vector already determines the entire graph. We therefore do not present it as a new encryption primitive, as a security contribution comparable to AEAD, or as a compression method. Its mathematical value is canonicalization: exact arithmetic, fixed indexing, deterministic expansion to $M(S_{n+1})$, reversible decoding, and explicitly characterized consistency checks.

8.2. When the Middle Graph Is Necessary

The middle graph is optional for transport but meaningful as an application-facing data structure when a downstream routine expects a weighted graph or adjacency matrix. Section 4.4 gives a concrete graph analytics scenario, and Section 7 gives its exact matrix instance. In such a workflow, the compact residues are transported, and the matrix is reconstructed at the consumer, avoiding quadratic transmission. If the receiver only needs the original bytes, the graph view is unnecessary, and direct AEAD is preferable. This distinction directly limits the scope of the claimed utility.

8.3. Security Boundary

The public encoder supplies no secrecy and no adversarial authentication. Proposition 5 quantifies only non-adversarial random-error detection for redundant clique checks. Proposition 6 is a composition sanity check; it confirms that adding a deterministic injective representation before an established AEAD does not create a new security claim. Modern AEAD terminology and properties are summarized in RFC 9771 [27]; concrete security continues to depend on the selected standardized primitive and its correct use.

8.4. Limitations and Alternatives

- (i) Ciphertext-only recovery yields y_i and the edge-vertex key rows, not x_i , z_i , or the plaintext by itself.
- (ii) Rounded-ciphertext recovery depends on conditioning and precision. Proposition 1 and Table 2 quantify this dependence for the tested model rather than asserting universal finite-precision recovery.
- (iii) The public graph encoding doubles the compact payload size because every residue is serialized in two bytes. It is slower than direct 16-bit widening and slower than direct AEAD.
- (iv) The $n \leq 256$ format bound is an engineering choice and not a graph-theoretic necessity. A larger field, another public tag schedule, CRT-based layout, or length-prefixed raw bytes can remove it.
- (v) The clique checks are not an error-correcting code. Their random-error detection probability is stated only for the explicit substitution model in Proposition 5.
- (vi) The dense matrix should not be transmitted unless an external graph-matrix interface specifically requires it; the compact residues are sufficient to reconstruct it.

These limitations are part of the design guidance rather than exceptions to it. The manuscript's central lesson is to separate a mathematically structured representation from the cryptographic primitive responsible for security.

9. Conclusions

This study performed a self-cryptanalysis of the authors' fully specified middle-graph matrix cipher. The central ciphertext row remains a unique linear combination of the leaf rows after right multiplication by the secret matrix. Consequently, one exact ciphertext deterministically reveals all leaf parameters y_i and the n secret-key rows indexed by edge vertices. The result was a partial (not total) ciphertext-only break: x_i , z_i , and the plaintext required additional information. Under key reuse, known plaintexts recovered the remaining key whenever the stacked adjacency matrices had full column rank. The attack was computationally modest, and the perturbation analysis and rounded-ciphertext experiments show how its practical success depends on precision and conditioning.

The canonical finite-field construction is therefore positioned only as a public middle-graph encoding. Its compact residue vector is the primary representation, and the dense graph matrix is an optional deterministic expansion for graph-domain consumers. Exact decoding, injectivity, and clique validation are representation guarantees; the quantified clique checks detect random corruption under a stated model but do not authenticate an adversary. Benchmarks show that direct encodings and direct AES-GCM are faster when graph structure is not needed. Confidentiality and integrity, when required, remain the responsibility of standard AEAD. The broader conclusion is that structural graph complexity can be mathematically useful, but cryptographic security must be established by explicit threat models, attacks, and standard security notions rather than by representation complexity alone.

Author Contributions: Conceptualization, S.N. and R.A.; methodology, F.Y. and M.S.A.; investigation, M.S.A. and M.W.A.; writing—original draft preparation, F.Y. and S.N.; writing—review and editing, R.A., M.S.A., and M.W.A.; funding acquisition, M.W.A. All authors have read and agreed to the published version of the manuscript.

Funding: This article was supported by the Faculty of Engineering and the Built Environment of the University of Johannesburg in South Africa.

Data Availability Statement: No external dataset was used. The supplementary code/scripts directory contains `reference_encoder.py` for the public encoder, decoder, matrix expansion, validation, graph-domain example, and baseline benchmark. It also contains `reviewer_experiments.py` for attack timing, finite-precision recovery, and AES-GCM comparison. Execution instructions and dependencies are given in `code/README.md` and `code/requirements.txt`. NumPy is required for the attack experiments; the Python cryptography package is required only for the AES-GCM benchmark. Fixed random seeds are included in the scripts.

Acknowledgments: M.W.A acknowledges the FEBE of the University of Johannesburg in South Africa for providing the postdoctoral fellowship.

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

References

1. Goldreich, O. *Foundations of Cryptography: Volume 1, Basic Tools*; Cambridge University Press: Cambridge, UK, 2001.
2. Stinson, D.R.; Paterson, M.B. *Cryptography: Theory and Practice*, 4th ed.; CRC Press: Boca Raton, FL, USA, 2018.
3. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography*, 3rd ed.; CRC Press: Boca Raton, FL, USA, 2020.
4. Ni, B.; Qazi, R.; Rehman, S.U.; Farid, G. Some graph-based encryption schemes. *J. Math.* **2021**, *2021*, 6614172. [[CrossRef](#)]
5. Razaq, A.; Alolaiyan, H.; Ahmad, M.; Yousaf, M.A.; Shuaib, U.; Aslam, W. A novel method for generation of strong substitution-boxes based on coset graphs and symmetric groups. *IEEE Access* **2020**, *8*, 75473–75490. [[CrossRef](#)]
6. Meenakshi, A.; Dhanushiya, S.; Mrsic, L.; Kalampakas, A.; Samanta, S. A multi layered encryption framework using intuitionistic fuzzy graphs and graph theoretic domination for secure communication networks. *Sci. Rep.* **2025**, *15*, 20992. [[CrossRef](#)] [[PubMed](#)]
7. Annamalai, M.; Mythreyi, O.; Mrsic, L.; Kalampakas, A.; Samanta, S. A fuzzy hypergraph-based framework for secure encryption and decryption of sensitive messages. *Mathematics* **2025**, *13*, 1049. [[CrossRef](#)]

8. Sowmya, K.K.; Srinivas, V. Cryptanalysis of some graph theory based encryption techniques. *WSEAS Trans. Math.* **2025**, *24*, 707–713. [\[CrossRef\]](#)
9. Sajjad, M.; Alqwaify, N.A. RGB image encryption using SPN with a novel block cipher over simple graph adjacency matrices and Galois fields. *Sci. Rep.* **2025**, *15*, 37488. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Biggs, N. *Algebraic Graph Theory*, 2nd ed.; Cambridge University Press: Cambridge, UK, 1993.
11. Diestel, R. *Graph Theory*, 5th ed.; Springer: Berlin/Heidelberg, Germany, 2017.
12. Priyadarsini, P.L.K. A survey on some applications of graph theory in cryptography. *J. Discret. Math. Sci. Cryptogr.* **2015**, *18*, 209–217. [\[CrossRef\]](#)
13. Deutsch, E.; Klavžar, S. M-polynomial and degree-based topological indices. *Iran. J. Math. Chem.* **2015**, *6*, 93–102. [\[CrossRef\]](#)
14. Gutman, I.; Redžepović, I.; Kizilirmak, G.Ö.; Kulli, V.R. Euler-Sombor index and its congeners. *Open J. Math. Sci.* **2025**, *9*, 141–148. [\[CrossRef\]](#)
15. Jahanbani, A.; Gutman, I. Exact variance-energy relations and optimal spectral bounds for graphs. *Open J. Math. Sci.* **2025**, *9*, 301–307. [\[CrossRef\]](#)
16. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [\[CrossRef\]](#)
17. Webster, A.F.; Tavares, S.E. On the design of S-boxes. In *Advances in Cryptology—CRYPTO '85 (Lecture Notes in Computer Science)*; Williams, H.C., Ed.; Springer: Berlin/Heidelberg, Germany, 1986; Volume 218, pp. 523–534. [\[CrossRef\]](#) [\[PubMed\]](#)
18. *NIST SP 800-38D*; Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. National Institute of Standards and Technology: Boulder, CO, USA, 2007.
19. *NIST SP 800-232*; Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions. National Institute of Standards and Technology: Boulder, CO, USA, 2025.
20. McGrew, D.A. An Interface and Algorithms for Authenticated Encryption (Request for Comments 5116). Internet Engineering Task Force. 2008. Available online: <https://www.rfc-editor.org/info/rfc5116/> (accessed on 1 January 2025).
21. Ghalavand, A.; Klavžar, S.; Tavakoli, M.; Yero, I.G. On mixed metric dimension in subdivision, middle, and total graphs. *Quaest. Math.* **2023**, *46*, 2517–2527. [\[CrossRef\]](#)
22. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [\[CrossRef\]](#)
23. Rosenfeld, A. Fuzzy graphs. In *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*; Zadeh, L.A., Fu, K.-S., Tanaka, K., Shimura, M., Eds.; Academic Press: Cambridge, MA, USA, 1975; pp. 77–95.
24. Mordeson, J.N.; Nair, P.S. *Fuzzy Graphs and Fuzzy Hypergraphs*; Physica: New York, NY, USA, 2001.
25. Bellare, M.; Namprempre, C. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology—ASIACRYPT 2000*; Lecture Notes in Computer Science; Okamoto, T., Ed.; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1976, pp. 531–545. [\[CrossRef\]](#) [\[PubMed\]](#)
26. Rogaway, P. Nonce-based symmetric encryption. In *Fast Software Encryption*; Lecture Notes in Computer Science; Roy, B., Meier, W., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3017, pp. 348–359. [\[CrossRef\]](#)
27. Bozhko, A. (Ed.) Properties of Authenticated Encryption with Associated Data (AEAD) Algorithms (RFC 9771). Internet Research Task Force. 2025. Available online: <https://www.rfc-editor.org/rfc/rfc9771.html> (accessed on 1 January 2025).
28. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 4th ed.; MIT Press: Cambridge, MA, USA, 2022.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.