Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 17** : 1–7.
ISSN-0037-8712
doi:10.5269/bspm.82383

A Graph-Based Framework for Recursive Self-Invertible Matrix Cryptography Using Complete Graphs (K_4)

V. Manoj Kumar and Uma Dixit

ABSTRACT: This paper proposes a novel symmetric encryption scheme that integrates algebraic graph theory with recursive matrix transformations. The method encodes plaintext into weighted complete graphs (K_4) and encrypts the resulting adjacency matrices using a dynamic sequence of self-invertible keys. Unlike static Hill ciphers, this approach utilizes a recursive function to generate unique keys for each data block, significantly enhancing security against cryptanalysis. We define the algebraic structure over a modular arithmetic system (mod 29), detail the recursive key generation algorithm, and provide three distinct numerical illustrations of the encryption and decryption protocols.

Keywords: Cryptography, algebraic graph theory, Self-Invertible Matrix, recursive encryption, complete graphs.

Contents

1. Introduction

Cryptography relies heavily on matrix transformations to obscure relationships between plaintext and ciphertext [2,4,15]. While classical methods like the Hill Cipher utilize static keys, they are vulnerable to known-plaintext attacks due to the linear nature of the transformation. Recent research has explored using graph theory—specifically labeling and adjacency matrices—to introduce non-linearity and complexity [5,6,10,11].

This work extends previous research on self-invertible matrix cryptosystems [1,7,10,12,13] by introducing a Recursive Self-Invertible Matrix protocol. In this system, the encryption key evolves for every block of text based on a deterministic recursive function derived from a seed matrix. This ensures that identical plaintext blocks result in different ciphertext blocks, providing a robust defense against frequency analysis.

2020 *Mathematics Subject Classification:* 94A60, 05C50, 11T71.
Submitted March 04, 2026. Published June 19, 2026.

2. Basic Graph Theory Concepts

To understand this method, we need to define a few simple terms [8,3,14,9]:

- **Graph:** A collection of dots (vertices) connected by lines (edges).
- **Complete Graph:** A graph where every dot is connected to every other dot directly.
- **Adjacency Matrix:** A grid of numbers representing the graph. If dot 1 is connected to dot 2, we put a number (weight) in the grid. If they are not connected, we put 0.

3. Preliminaries and Symbolic Framework

3.1. Encoding Scheme

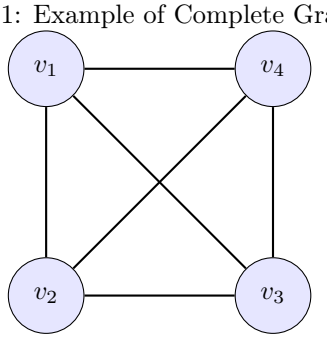
We utilize a modulo 29 arithmetic system to map characters to numerical values. The encoding table is defined as follows:

- Alphabets A through Z map to integers 1 through 26.
- The Space character maps to 27.
- The Dummy character '&' maps to 28.
- All arithmetic operations are performed modulo 29.

3.2. Complete Graph K_4 Representation

Let a message block consist of 6 characters. These characters are mapped to the 6 edges of a complete graph K_4 with vertices $\{v_1, v_2, v_3, v_4\}$.

Figure 1: Example of Complete Graph K_4



Let the numerical values of the block be $\{w_1, w_2, w_3, w_4, w_5, w_6\}$. The mapping to edges is defined as:

$$(v_1, v_2) \to w_1 \quad (v_1, v_3) \to w_2 \quad (v_1, v_4) \to w_3$$
$$(v_2, v_3) \to w_4 \quad (v_2, v_4) \to w_5 \quad (v_3, v_4) \to w_6$$

The adjacency matrix A of this weighted graph is a symmetric 4×4 matrix where $A_{ij} = A_{ji}$ is the weight of the edge connecting v_i and v_j.

3.3. Self-Invertible Matrix Construction

A matrix K is self-invertible (or an involutory matrix) if $K^2 \equiv I \pmod{n}$. To create a large key matrix of size $n \times n$, we only need to create a smaller random matrix of size $\frac{n}{2} \times \frac{n}{2}$ [1]. Where I is an Identity matrix.
In this paper we construct a 4×4 self-invertible matrix K_{44} from a given 2×2 seed matrix K_{22}, using the block construction method:

$$K_{44} = \begin{bmatrix} -K_{22} & I - K_{22} \\ I + K_{22} & K_{22} \end{bmatrix} \pmod{29} \tag{3.1}$$

It is easily verified that $K_{44}^2 = I$, regardless of the choice of K_{22}.

4. Proposed Algorithm

4.1. Recursive Key Generation

To ensure dynamic encryption, the seed matrix K_{22} changes for each block i. Let the initial seed be $K_{22_0} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. The recursive update rule for the i-th block (where $i \geq 0$) is defined as:

$$K_{22_i} = \begin{pmatrix} a+i & b-i \\ c-i & d+i \end{pmatrix} \pmod{29} \tag{4.1}$$

For each block i, a unique self-invertible key K_{44_i} is generated using K_{22_i} and Eq. (3.1).

4.2. Encryption Process

1. **Block Division:** Divide plaintext into blocks of 6 characters. If the last block has fewer than 6 characters, pad with '&' (value 28).
2. **Graph Conversion:** Convert each block i into weights and construct the adjacency matrix A_i.
3. **Key Generation:** For each block i, compute K_{22_i} and the corresponding K_{44_i}.
4. **Transformation:** Compute the cipher matrix C_i:

$$C_i = K_{44_i} \times A_i \times K_{44_i} \pmod{29}$$

5. **String Generation:** Flatten C_i (row-wise) into a string of numbers.
6. **Transmission:** The final output format is:

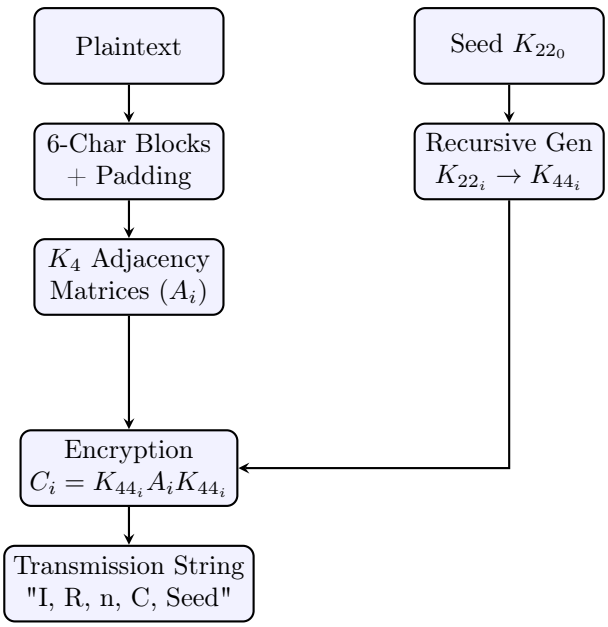
"I, R, n, < Encrypted Strings >, < Seed Elements >"

Where I is the number of blocks, $R = 1$ (indicating recursive mode) & $R = 0$ (indicating non-recursive mode), and $n = 4$ (matrix size).

4.3. Implementation Data Flow

The final transmission string for any message follows the structure defined as follows

Figure 2: Encryption Data Flow



4.4. Decryption Process

Decryption is the inverse of encryption. However, since the key matrices are self-invertible, the inverse key is the key itself ($K^{-1} = K$).

1. **Receive Data:** Extract the seed matrix K_{22_0} and the cipher matrices $C_0, C_1, ..., C_n$.
2. **Regenerate Keys:** Using the single shared seed K_{22_0}, the receiver independently calculates the sequence $K_{22_1}, K_{22_2}, ...$ using Eq. (4.1).
3. **Inverse Transformation:** For each block i:
$$A_i = K_{44_i} \times C_i \times K_{44_i} \pmod{29}$$
4. **Decoding:** Extract edges from the upper triangle of A_i to retrieve weights, convert to characters, and discard padding.

5. Worked Illustrations

Example 5.1 Example 1 :Single Block Illustration:
Plaintext: "GRAPH " (Note: Space at end)
Seed K_{22_0} : $\begin{pmatrix} 25 & 15 \\ 19 & 10 \end{pmatrix}$
Encryption:

1. **Encoding:** $G = 7$, $R = 18$, $A = 1$, $P = 16$, $H = 8$, $Space = 27$.
2. **Adjacency Matrix** (A_0)**:** Constructed from weights $\{7, 18, 1, 16, 8, 27\}$.
3. **Key** (K_{44_0})**:** Generated from K_{22_0}.
4. **Cipher** (C_0)**:** $C_0 = K_{44_0} A_0 K_{44_0} \pmod{29}$.

Decryption: The receiver possesses C_0 and the seed K_{22_0}.

1. **Key Regeneration:** The receiver computes K_{44_0} using the known seed.
2. **Matrix Recovery:** $A_0 = K_{44_0} C_0 K_{44_0} \pmod{29}$. Since K_{44_0} is self-invertible, this operation reverses the encryption perfectly.
3. **Extraction:** The receiver reads the upper triangle of A_0: Values $\{7, 18, 1, 16, 8, 27\} \rightarrow$ "GRAPH ".

Example 5.2 Example 2 :Three-Block Scenario (Comprehensive):
Plaintext: "SECURE DATA SET" (15 characters)
Structure:

- Block 0: "SECURE" (6 chars)
- Block 1: " DATA " (6 chars, includes spaces)
- Block 2: "SET" (3 chars) $\rightarrow$ Pads to "SET&&&"

Seed K_{22_0} : $\begin{pmatrix} 25 & 15 \\ 19 & 10 \end{pmatrix}$
Phase 1: Recursive Key Generation

- **Block 0 Key** $(i = 0)$**:** Generates K_{44_0} from K_{22_0}.
- **Block 1 Key** $(i = 1)$**:** Apply recursion: $a + 1$, $b - 1$, $c - 1$, $d + 1$.

$$K_{22_1} = \begin{pmatrix} 25+1 & 15-1 \\ 19-1 & 10+1 \end{pmatrix} = \begin{pmatrix} 26 & 14 \\ 18 & 11 \end{pmatrix} \pmod{29}$$

  Generates K_{44_1}.

- **Block 2 Key** $(i = 2)$**:** Apply recursion: $a + 2$, $b - 2$, $c - 2$, $d + 2$.

$$K_{22_2} = \begin{pmatrix} 25+2 & 15-2 \\ 19-2 & 10+2 \end{pmatrix} = \begin{pmatrix} 27 & 13 \\ 17 & 12 \end{pmatrix} \pmod{29}$$

  Generates K_{44_2}.

Phase 2: Encryption (Sender)

1. Block 0 ("SECURE"): Compute A_0. Encrypt: $C_0 = K_{44_0} A_0 K_{44_0}$.
2. Block 1 (" DATA "): Compute A_1. Encrypt: $C_1 = K_{44_1} A_1 K_{44_1}$.
3. Block 2 ("SET&&&"): Compute A_2. Encrypt: $C_2 = K_{44_2} A_2 K_{44_2}$.

Transmission String: "3, 1, 4, Flat C_0, Flat C_1, Flat C_2, 25, 15, 19, 10".
Phase 3: Decryption (Receiver) The receiver extracts the seed "25, 15, 19, 10".

1. **Block 0 Decryption:** Use seed to form K_{44_0}. Calculate $A_0 = K_{44_0} C_0 K_{44_0} \pmod{29}$. Extract weights $\rightarrow$ "S, E, C, U, R, E".
2. **Block 1 Decryption:** Compute K_{22_1} by adding index 1 to diagonals and subtracting 1 from off-diagonals of the seed. Form K_{44_1}. Calculate A_1 and extract weights $\rightarrow$ " , D, A, T, A, ".
3. **Block 2 Decryption:** Compute K_{22_2} by adding index 2 to diagonals and subtracting 2 from off-diagonals of the seed. Form K_{44_2}. Calculate A_2. Extract weights "S, E, T, &, &, &". Post-processing: The receiver identifies '&' (28) as dummy characters and removes them.

Final Result: "SECURE DATA SET".

Example 5.3 Example 3 : Padding Detail:
Plaintext: "KEY" (3 chars)
Process: Since the text is less than 6 characters, it is treated as a single block requiring padding.

- Block: "KEY&&&".
- Values: $K(11)$, $E(5)$, $Y(25)$, &(28), &(28), &(28).
- Adjacency Matrix (A_0):

$$A_0 = \begin{pmatrix} 0 & 11 & 5 & 25 \\ 11 & 0 & 28 & 28 \\ 5 & 28 & 0 & 28 \\ 25 & 28 & 28 & 0 \end{pmatrix}$$

- Encryption/Decryption: Follows the standard single-block procedure ($i = 0$).

6. Conclusion

This paper presented a recursive matrix cryptosystem based on complete graph adjacency matrices. By integrating a time-varying key generation mechanism (K_{22_i}), the system overcomes the static key vulnerabilities of traditional algebraic ciphers. The use of self-invertible matrices simplifies the protocol by eliminating the need for matrix inversion during decryption, making it computationally efficient for both sender and receiver.

References

1. Acharya, B. et al. (2007). Generating self-invertible matrices for Hill Cipher.
2. Diffie, W., Hellman, M. (1976). New directions in Cryptography.
3. Arumugam, S., Ramachandran, S. (2015). Invitation to Graph Theory.
4. Koblitz, N. (1994). A Course in Number Theory and Cryptography.
5. Yamuna, M., et al. Encryption using Graph theory and Linear algebra.
6. Nandhini, R., et al. (2018). A Graph Theory Approach on Cryptography.
7. Acharya, B., et al. (2009). Image encryption using advanced Hill Cipher algorithm.
8. Narsingh Deo (2004). Graph Theory with Applications to Engineering and Computer Science.
9. Uma Dixit, Cryptography a Graph theory approach, International journal of Advance Research in Science and Engineering, 6(01), 2017.
10. P. Mohan, K. Rajendran, A. Rajesh, Journal of Algebraic statistics 13(3), 2022. https://publishoa.com/index.php/journal/article/view/816, pp.1821-1826.
11. P. Mohan, K. Rajendran, A. Rajesh, Indian Journal of Science and Technology 15(44), 2351-2355, 2022.
12. Mohan P, Rajendran K, Rajesh A. An encryption Technique using the adjacency matrices of certain graphs with a self-invertible key matrix, E3S Web of Conf, Volume 376, 01108(2023) https://doi.org/10.1051/e3sconf/202337601108.
13. Mohan P, Rajendran K, Rajesh A. Enhancing Computational Performance of Minimal Spanning Tree of Certain Graphs Based Enciphering Technique Using Self-Invertible Key Matrix, Journal of Aeronautical Materials(1005-5053), Vol 43, Issue-01(2023), pp 359-371, https://www.hkclxb.cn/article/view/2023/359.html.
14. A. Rosenfeld, *Fuzzy Graphs*, Academic Press, (1975).
15. W. Stallings, *Cryptography and Network Security*, Pearson, (2023).

Acknowledgments

We thank the **Department of Mathematics,Osmania University** and **Vivekananda Government Degree College, Vidyanagar**for providing the academic and computational resources necessary for this research.

V. Manoj Kumar
Department of Mathematics,
Vivekananda Govt.Degree College
Vidyanagar, Hyderabad,
India.
E-mail address: vmyasho@gmail.com

and

Uma Dixit
Department of Mathematics,
University College Of Science
Osmania University,Hyderabad,
India.
E-mail address: umadixit@osmania.ac.in