Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 27 (2024), No. 7, pp. 2155–2167
DOI : 10.47974/JDMSC-2088

Enhancing cryptographic robustness through error-correcting codes derived from network graphs

Gayathri Ananthakrishnan *
School of Computer Science Engineering and Information Systems
Vellore Institute of Technology
Vellore 632014
Tamil Nadu
India

Hayder M. A. Ghanimi †
Department of Information Technology
College of Science
University of Warith Al-Anbiyaa
Karbala 56001
Iraq

and

Department of Computer Science
College of Computer Science and Information Technology
University of Kerbala
Karbala 56001
Iraq

P. Pushpa §
Department of Cyber Security
Easwari Engineering College
Chennai 600089
Tamil Nadu
India

* *E-mail:* gayathri.a@vit.ac.in (Corresponding Author)
† *E-mail:* hayder.alghanami@uowa.edu.iq
§ *E-mail:* ppushpacse88@gmail.com

T. K. Rama Krishna Rao [‡]
Department of Computer Science and Engineering
Koneru Lakshmaiah Education Foundation
Vaddeswaram 522502
Andhra Pradesh
India

M. Vedaraj [@]
Department of Computer Science and Engineering
RMD Engineering College
Kavaraipettai 601206
Tamil Nadu
India

Sudhakar Sengan [#]
Department of Computer Science and Engineering
PSN College of Engineering and Technology
Tirunelveli 627152
Tamil Nadu
India

Pankaj Dadheech [$]
Department of Computer Science and Engineering
Swami Keshvanand Institute of Technology, Management & Gramothan (SKIT)
Jaipur 302017
Rajasthan
India

Abstract

Secure data transmission is essential for securing sensitive data generated in huge volumes in today's growing digital era. In today's growing digital world, secure data transfer is essential for securing private information generated in enormous volumes. Data security strategies, called Cyber Security Systems (CSS), are susceptible to errors and other risks that might damage the data's integrity. While standard Error-Correcting Codes (ECCs) function well for more standard communication errors, they cannot meet the complex rules of cryptographic functions. In order to boost the CSS size, this article provides a detailed approach that develops ECCs from network graph parameters. The error flexibility, computational speed, and use of bandwidth power of the CSS have been designed to be

[‡] *E-mail:* **ramakrishnatk@gmail.com**
[@] *E-mail:* **vedaraj84@gmail.com**
[#] *E-mail:* **sudhasengan@gmail.com**
[$] *E-mail:* **pankajdadheech777@gmail.com**

addressed by the graph-based ECCs. The investigation proposes an approach that analyses network graphs and employs information to develop ECCs, which are then integrated into the existing CSS. Compared with standard models, the one recommended performed more successfully when assessed under a range of error scenarios and cryptanalytic attacks.

Subject Classification: *34A06.*

Keywords: *Mathematical modeling, Ordinary differential equations, A duopoly economy, Market share.*

1. Introduction

Communicating data reliably is essential in the rapidly developing online environment. The financial sector, medical services, and encrypted messaging use CSS to secure data [1-2]. Despite significant improvements, CSS is vulnerable to errors and attacks that breach data security and integrity [3]. Traditional Error-Correcting Codes (ECCs) like Hamming and Reed-Solomon Codes (HRSC) are a highly prevalent method for TE avoidance [4]. Classic ECCs could not handle CSS operations with intricate error codes and significant cryptanalytic attacks [5]. Such models also consume data and computational energy in situations with limited resources. This study involves creating and implementing network graph-derived ECCs to enhance CSS reliability [6]. Network graphs provide ECCs that avoid conventional TE and complex cryptographic errors. [7-8] ECCs that enhance error robustness, computational demand, and bandwidth utilisation employ network graph features, including communication, vertex degree, and cycle structures. [9-10]

The objectives of this study are:

(a) To build ECCs with network graph attributes,

(b) To integrate these ECCs into existing CSS effectively and

(c) To evaluate their performance across multiple scenarios.

The paper is organized as follows: Section 2 presents the background of the work, Section 3 presents the methodology, Section 4 presents the experiment analysis, and Section 5 presents the conclusion and future work.

2. Background

2.1 *Overview of ECC*

ECCs are algorithms or sequences of codes used to implement Error-Detection (ED) and Error-Correction (EC) in communication systems. They add redundancy to the original data, enabling the ED and EC at the receiver's end. The core functions of ECC can be expressed as follows: EQU (1) and EQU (2)

- **Encoding Function:**

$$C = \text{encode}(M) \tag{1}$$

where M is the original message, C is the encoded message containing additional redundant bits.

- **Decoding Function:**

$$\hat{M} = decode(C) \tag{2}$$

where C is the received coded message, hypothetically containing errors and $\hat{M}$ is the decoded message, representing the estimate of the original message.

The primary role of ECC is to ensure data integrity and reliability across noisy channels. Practically, this means correcting errors up to a certain number of bits. This capability is quantified by the code's distance properties:

- **Hamming Distance (d):** It represents the minimum number of differing bits between 2 valid codewords in the code set. It determines the ED and EC capabilities of the code EQU (3) and EQU (4).

$$\text{Number of } ED = d - 1 \tag{3}$$

$$\text{Number of } EC = \left\lfloor \frac{d-1}{2} \right\rfloor \tag{4}$$

- **Traditional Generation of ECC:** ECC is traditionally generated using several statistical methods, primarily linear block codes and cyclic codes:
- **Linear Block Codes:** These codes are defined using matrices. A linear block code of length n and dimension k is generated by a generator matrix G of size $k \times n$. The encoding process is a linear transformation, EQU (5).

$$C = M \cdot G \tag{5}$$

where M is a k-bit message, and C is the n-bit codeword.

- **Cyclic Codes:** Cyclic codes are a subtype of linear block codes where each cyclic shift of a codeword results in another codeword. A popular example is the Cyclic Redundancy Check (CRC), which is used primarily for error detection.

2.2 *Introduction to Network Graphs*

Network graphs are mathematical structures that are used to model relationships between objects. A network graph G can be represented as a tuple $G = (V, E)$, where:

- V is a set of vertices or nodes representing the objects in the network.
- E is a set of edges representing the connections or relationships between these objects.

Each edge in 'E' can be denoted as an ordered or unordered pair (u, v) depending on whether the graph is directed (relationships have a direction) or undirected. Additionally, graphs can be weighted, where each edge (u, v) has an associated weight 'w', representing the strength or capacity of the connection.

a. *Properties Relevant to ECC*

- **Connectivity:** This property indicates whether every pair of vertices in a graph is connected by a path. In ECC, high connectivity enhances the code's ability to correct multiple distributed errors.
- **Graph Diameter:** The diameter of a graph is the longest and shortest path between any pair of vertices. A smaller diameter implies that information (errors) can propagate rapidly across the network, affecting the design and efficiency of ECC.
- **Vertex Degree:** The degree of a vertex is the number of edges connected to it. For network-wide EC, ECC nodes with greater levels are required.
- **Cycles and Cycle Bases:** A graph's cycles include sets of edges that begin and end at a single vertex and are not recurring. All cycles can be generated by cycle bases, which assists in developing cyclic ECCs that use cyclical attributes for EC.

3. Proposed Methodology

3.1 *Graph-Theoretic Approach (GTA) to ECC*

The process starts with a network graph $G = (V, E)$, where 'V' signifies vertices or nodes, and 'E' represents edges. Vertex and edge indicate the structure of data elements, and the graph's layout defines how they interact. It builds ECCs leveraging connectivity, cycles, and vertex degrees. ECCs are calculated from network graphs using graph-theoretic procedures. The initial algorithm is the Welsh-Powell Algorithm (WPA). This method uses degree and colour to confirm that no adjacent vertices share an identical colour. This eliminates data point confusion and is EQU (6):

$$\forall v \in V, Color(v) = \min\{c \mid c \notin \{Color(u) \mid u \in Adj(v)\}\} \tag{6}$$

Horton's Algorithm (HA) detects and encodes data over graph cycles to add duplication and provide several reference points for each data set. The cycles are identified and applied as follows EQU (7):

$$\text{Cycle Basis} = \{C \mid C \text{ forms a cycle in } G\} \tag{7}$$

Kruskal's algorithm (KA) is applied to optimise any redundant system by generating a minimum spanning tree (MST). Weight-based edge selection minimises time and improves network connectivity, EQU (8).

$$\text{MST} = \{e \in E \mid \text{adding } e \text{ maintains acyclicity}\} \tag{8}$$

Algorithm for ECC Generation from Network Graphs

Input: $G = (V, E)$, where V is the set of vertices and E is the set of edges.

Output: An ECC scheme derived from G.

Step 1: Graph Coloring Using Welsh-Powell:

- Sort all vertices 'V' in descending order based on their degree.
- Start an empty list of colors and assign the minor color to each vertex 'v'.
- **For Each vertex 'v' in the sorted list,** Assign to 'v' the lowest numbered color that has not been used by its adjacent vertices, EQU (9).

$$\text{Color}(v) = \min\{c \mid c \notin \{\text{Color}(u) \mid u \in \text{Adj}(v)\}\} \tag{9}$$

- Continue until all vertices are colored.

Step 2: Cycle-Based Coding Using HA

- Identify all simple cycles in '*G*' using Depth-First Search (DFS) or other cycle-finding techniques.
- From the identified cycles, construct a cycle basis, which includes minimal and fundamental cycles for ECC.
- Encode data along these cycles, utilizing the redundancy of overlapping cycles to implement error correction.

Step 3: Minimum Spanning Tree (MST) Construction Using KA

- Create a set '*S*' to hold the edges of the MST.
- Sort all edges '*E*' in ascending order by weight.
- For Each edge $e = (u, v)$ in the sorted edge list: If 'u' and 'v' are in different components (use the union-find data structure to check), add 'e' to 'S'.
- Use union-find to merge the components of 'u' and 'v'.
- The edges in '*S*' now form the MST, which outlines the minimum connections required for effective ECC.

Step 4: Integration and Testing:

- Integrate the ECC derived from the above steps into the target communication system.
- Test the integrated system for error correction capability and efficiency under various conditions to ensure robustness and performance.

Step 5: End Algorithm

4. Experiment Simulation

The simulation model simulates communication scenarios, as shown in Table 1, to evaluate the ECCs in many conditions. It operates in a computing environment replicating network communication systems with variable signal strengths, noise levels, and intrusion patterns.

A) *Model Description*

- **Data Generation (*D*):** Generates random data sets represented as '*D*', each containing n bits of data.

- **Encoding Process (*E*):** Encodes data '*D*' using the ECCs derived from graph-theoretic algorithms, producing encoded data *E*(*D*).
- **Transmission Channel (*C*):** Transmits the encoded data *E*(*D*) through a simulated noisy channel '*C*' that introduces random (e_r), burst (e_b), and systematic (e_s) Errors.
- **Decoding and Error Correction (D_c):** Decodes the received data 'D_r' using the same ECC, attempting to correct errors introduced during transmission. The output is the corrected data. 'D_c'.

Table 1

Simulation Scenarios and Parameters

Scenario	Data	Transmission Size (bits)	Error Type	Error Rate	Channel Conditions	Repetition of Trials
1	Binary	10,000	Random	1%	Low Noise	30
2	Binary	10,000	Burst	2%	Medium Noise	30
3	Binary	50,000	Systematic	0.5%	High Noise	30
4	Textual	10,000	Random	3%	High Interference	30

B) *Baseline Models for Comparison*

- **Hamming Code (HC):** HC is a binary linear code capable of detecting up to 2-bit errors and correcting 1-bit errors per transmission block. The typical setup involves parameters that define message length and redundancy bits, structured around the formula $(2^r - 1,\ 2^r - 1 - r)$.
- **Reed-Solomon Code (RSC):** RSC is adept at correcting multiple symbol errors within a data block. Standard configurations for these codes use parameters like (255, 223) in bytes, including 32 bytes of parity.

C) *Metrics for comparison:*

- **ECC** measures the maximum EC in each data block, EQU (10).

$$ECC = \text{Number of corrected errors per data block} \quad (10)$$

- **Computational Overhead (CO)** quantifies the additional computational resources required for cryptographic functions, EQU (11).

$$CO = \frac{T_{\text{encoded}} - T_{\text{raw}}}{T_{\text{raw}}} \tag{11}$$

- **Data Throughput (DT)** indicates the rate at which data is successfully transmitted and decoded, EQU (12).

$$DT = \frac{D_{\text{corrected}}}{T_{\text{transmission}}} \tag{12}$$

- **Bandwidth Efficiency (BE)** evaluates how effectively the ECC uses the available bandwidth, considering the overhead for EC and EQU (13).

$$BE = \frac{D_{\text{net}}}{D_{\text{gross}}} \tag{13}$$

Figure 1 and Table 2 display the performance of the proposed ECC derived from network graphs against traditional HA and RSA models for their error correction capability against different scenarios. In all the noise environments, the proposed ECC model corrects more errors than the HC and the RSC. In high noise (HN), burst error (BuR), and random error (RE) conditions, the proposed ECC model significantly outperformed other

Table 2

ECC Capability Comparison

Scenario	Proposed ECC (EC)	HC (EC)	RSC (EC)
1	4	1	2
2	10	1	5
3	15	2	7
4	5	1	3

Figure 1

EC Analysis

Table 3
CO Comparison

Scenario	Proposed ECC (Time ms)	HC (Time ms)	RSC Code (Time ms)
1	12	8	20
2	15	8	25
3	18	10	30
4	14	9	22

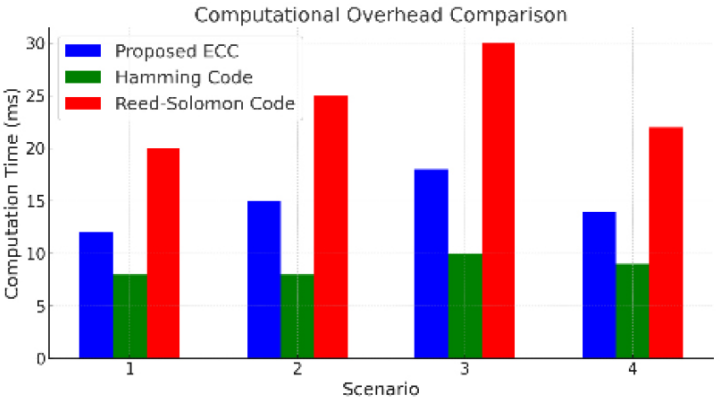

Figure 2
CO Graphical analysis

models by correcting a higher number of errors (15). The next better-performing model was the RSC, which showed better performance than the HC for all scenarios.

The Table 3 and the Figure 2 displays the comparison results for CO. The proposed ECC model displays more CT than the HC but less than the RSC in all scenarios. To be specific, the proposed model, in low noise environments, has exposed CT that is 50% higher than HC's but lower than RSC. This trend continues in HN, BuR, and RE scenarios, with the proposed ECC maintaining a middle-ground performance. Comparatively, the RSC was the model with higher Computation Time (CT) than the other two models, and the HC showed better CT for all test scenarios.

The DT comparison is shown in Table 4 and Figure 3. The proposed ECC have consistently demonstrated better DT by outperforming the RSC and HC in all scenarios. For instance, in a low noise scenario, the proposed

Table 4

DT Comparison

Scenario	Proposed ECC (Mbps)	HC (Mbps)	RSC (Mbps)
1	98	81	72
2	63	52	41
3	59	45	36
4	78	61	50

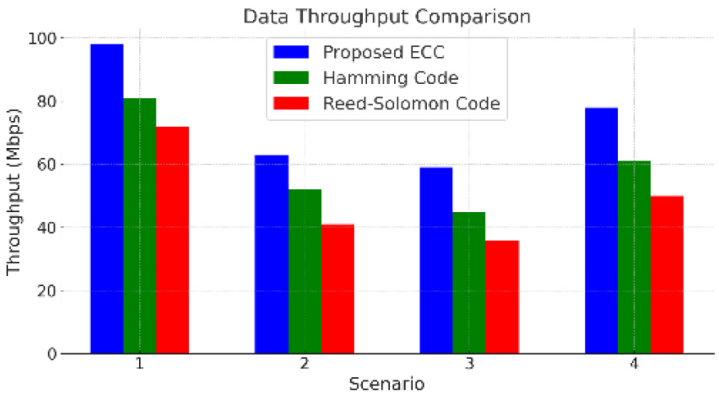

Figure 3

DT Comparison

ECC achieves 98 Mbps, which is higher than RSC's 72 *Mbps* and HC's 81 *Mbps*. This pattern is found to continue across all scenarios, such as HN, BuR, and RE conditions, where the proposed ECC maintains better DT than other models. Following the proposed model, the HC model showed better DT across all scenarios than the RSC, with the highest throughput of 81 *Mbps* for low-noise scenarios.

Figure 4 and Table 5 display the BE comparison of all models for different scenarios. In all scenarios, the proposed ECC ranks the top against the HC and RSC regarding efficiency. A low HN scenario shows a BE of 95%, slightly higher than HC's 92% but significantly better than RSC's 85%. This trend continues across HN, BuR, and RE scenarios, where the proposed ECC consistently maintains higher efficiency than both models. In the other two models, the HC model again showed better efficiency against the RSC model for all the scenarios, ranging between 92% and 85%.

Table 5
Bandwidth Efficiency Comparison

Scenario	Proposed ECC (Efficiency %)	HC (Efficiency %)	RSC (Efficiency %)
1	95%	92%	85%
2	90%	88%	80%
3	88%	85%	75%
4	93%	90%	82%

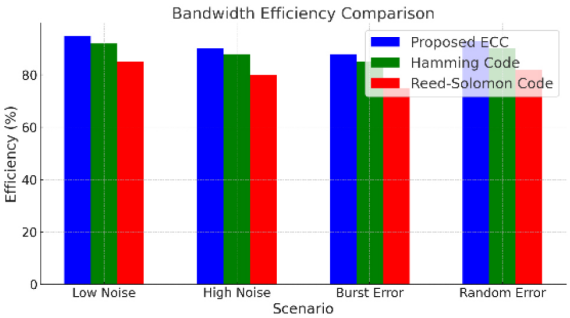

Figure 4
BE Analysis

5. Conclusion

Cryptographic systems that integrate Error-Correcting Codes (ECC) derived from network graphs are the primary topic of the present paper. The study's primary objective is to improve the system's capacity to endure easy and significant errors. The recommended ECC model performed better by reliably reducing EC and ED and increasing BE in all evaluated cases than other standard models regarding EC capabilities. Modern CSS algorithms which require high levels of data integrity and security may benefit from their improved performance in circumstances with strong resource limits and high error rates.

References

[1] R. A. Teubner and J. Stockhinger, "Literature review: Understanding information systems strategy in the digital age," *The Journal of Strategic Information Systems*, vol. 29, no. 4, pp. 101642 (2020).

[2] A. I. Newaz, M. H. A. Rahman, M. A. M. T. Z. Rahman, and A. R. A. Rahman, "A survey on security and privacy issues in modern healthcare systems: Attacks and defenses," *ACM Transactions on Computing for Healthcare*, vol. 2, no. 3, pp. 1-44 (2021).

[3] J. P. A. Yaacoub, K. M. A. Al-Muhtadi, and M. S. H. Al-Bayati, "Cyber-physical systems security: Limitations, issues and future trends," *Microprocessors and Microsystems*, vol. 77, no. 103201 (2020).

[4] M. A. Ghanimi, R. Venkatesan, and R. R. D. S. Shivaraj, "Chebyshev polynomial approximation in CNN for zero-knowledge encrypted data analysis," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 27, no. 2-A, pp. 203–214 (2024).

[5] P. Vidya Sagar, S. R. Raghunathan, and M. A. R. K. Marimuthu, "Secure multi-party computation in deep learning: Enhancing privacy in distributed neural networks," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 27, no. 2-A, pp. 249–259 (2024).

[6] S. H. Nowfal, R. A. R. Alqurashi, and Y. A. N. B. Alghamdi, "Advancing viscoelastic material modeling: Tackling time-dependent behavior with fractional calculus," *Journal of Interdisciplinary Mathematics*, vol. 27, no. 2, pp. 307–316 (2024).

[7] P. Vidya Sagar, S. R. Raghunathan, and M. A. R. K. Marimuthu, "Utilizing stochastic differential equations and random forest for precision forecasting in stock market dynamics," *Journal of Interdisciplinary Mathematics*, vol. 27, no. 2, pp. 285–298 (2024).

[8] G. R. K. Rao, S. R. Raghunathan, M. A. R. K. Marimuthu, and A. K. S. Ravi, "Enhanced security in federated learning by integrating homomorphic encryption for privacy-protected, collaborative model training," *Journal of Discrete Mathematical Sciences and Cryptography,* vol. 27, no. 2-A, pp. 361–370 (2024).

[9] K. Raguru Jaya, P. Vidya Sagar, and S. R. Raghunathan, "Security and privacy concerns in social networks: Mathematically modified metaheuristic-based approach," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 27, no. 2-A, pp. 371–382 (2024).

[10] S. Sudhakar and S. Chenthur Pandian, "Hybrid cluster-based geographical routing protocol to mitigate malicious nodes in mobile ad hoc network," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 21, no. 4, pp. 224–236 (2016).

Received March, 2024