10.48047/jocaaa.2024.33.07.127

IMPROVING ERROR TOLERANCE IN GRAPH-THEORETIC CRYPTOGRAPHIC PROTOCOLS THROUGH FUZZY CUBE DIFFERENCE LABELING

[1]M. Abirami, [2] S. Jackson

[1]Research Scholar (23212232092001), [2] Assistant Professor
[1, 2] P.G and Research Department of Mathematics, V.O. Chidambaram College, Thoothukudi–628008.
Affiliated to Manonmaniam Sundaranar University, Tirunelveli–627012, Tamil Nadu, India.[2]
[1]abiramimaharajan96@gmail.com , [2]jmjjack2008@gmail.com

Received: 09.09.2024 Accepted: 27.10.2024 Published: 20.11.2024

Abstract: This study presents Fuzzy Cube Difference Labeling (FCDL) as a novel cryptographic framework that leverages fuzzy vertex memberships and cube-difference edge values to enhance nonlinearity and entropy, thereby strengthening resistance against brute-force and structural attacks. Using complete bipartite graphs to map fuzzy cube difference values to alphabets, the proposed encryption and decryption algorithms demonstrate feasibility and effectiveness through MATLAB and Python implementations, highlighting FCDL's potential to advance secure graph-based post-quantum cryptography.

Keywords phrases: Fuzzy Cube Difference Labeling, Fuzzy Cube Difference Graph, Cipher Text, Encryption, Decryption, Cryptography.

2020 Mathematics Subject Classification: 05C78, 94A60, 68P25, 03E72.

1. Introduction

Cryptography is the science of secure communication, involving the encryption of plaintext into Ciphertext and decryption back into plaintext [7]. Modern applications include digital signatures, authentication, and secure protocols. Graphs provide a natural way to model such problems. In graph theory, labeling assigns values to vertices and edges according to specific rules. Since A. Rosa's first work in 1966 [8], many labeling techniques have been studied, with applications in frequency assignment, coding, and networks [3].Earlier works applied graph labeling methods like graceful and magic labeling to cryptography [1,10]. In this paper, we extend this idea using Fuzzy Cube Difference Labeling (FCDL), where vertices are assigned fuzzy membership values and edges are labeled by cubic differences. This labeling introduces non-linearity and fuzziness, making it suitable for secure encryption and decryption.

10.48047/jocaaa.2024.33.07.127

2. Preliminaries

This section introduces fundamental concepts and definitions essential for the discussion of specialized graph labelings. A clear understanding of these preliminaries is crucial for the development and analysis of the new labeling techniques presented in this paper.

Definition 2.1. The process of giving each graph's vertices and edges a unique identity is known as graph labeling.

Definition 2.2. Let $G = (X, Y)$ be a graph. If there exists an injection $\varphi: X \to \{0,1,\ldots, p-1\}$ such that the induced function $\varphi *: Y \to N, \varphi * (xy) = |\varphi(x)3 - \varphi(y)3|$ for each $xy \in Y$, is distinct (i.e., all values are different), then G is said to admit a **Cube Difference Labeling (CDL)**. A **Cube Difference Graph (CDG)** is a graph that admits a Cube Difference Labeling. [9].

Definition 2.3. A graph $G = (\sigma, \mu)$ is said to be a **fuzzy labeling graph**, if $\sigma: V \to [0,1]$ and $\mu: V \times V \to [0,1]$ is bijective such that the membership value of edges and vertices are distinct and $\mu(u, v) < min\{\sigma(u), \sigma(v)\}$ for all $u, v \in V$ [4, 5].

Definition 2.4. A **Fuzzy Cube Difference Labeling (FCDL)** is a fuzzy labeling graph $G = (V, E)$ that meets the following conditions: $\mu(u, v) = |[\sigma(u)]3 - [\sigma(v)]3|$ for each $(u, v) \in E$ is distinct. A **Fuzzy Cube Difference Graph (FCDG)** is a graph that supports FCDL. The labeling is shown in Figure 1 as an example.

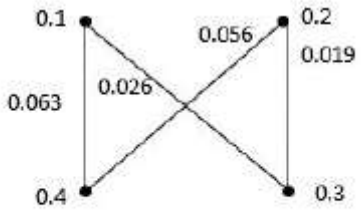


Figure 1: FCDL of $K_{2,2}$

3. METHODOLOGY

Here, we outline the encryption and decryption techniques employing Fuzzy Cube Difference Labeling (FCDL).

10.48047/jocaaa.2024.33.07.127

We consider the **complete bipartite graph $K_{n,n}$ with $n \geq 5$**, so that the resultant matrix will have at least 25 edge values. These distinct values can be mapped to the 26 letters of the English alphabet (assigning one value for each alphabet and reserving zero or the smallest value for the last alphabet). For languages with larger alphabet sets, such as Tamil, which has 247 symbols, we may consider a complete bipartite graph with at least 15 vertices on each side. In such cases, manual encryption/decryption becomes more difficult, and hence an automated program is provided.

In this algorithm, edge weights are determined based on the fuzzy cube difference labeling, and the encryption procedure proceeds as follows.

3.0.1 Encryption Algorithm (FCDL-based)

Input: Plain text message, integer $n \geq 5$.

Output: Cipher text message.

Step 1. Consider the complete bipartite graph $K_{n,n}, n \geq 5$.

Step 2. Assign distinct fuzzy membership values $\sigma(v) \in [0,1]$ to each vertex.

Step 3. For every edge $(u, v) \in E(K_{n,n})$, compute the fuzzy cube difference label:

$$\mu(u, v) = |[\sigma(u)]^3 - [\sigma(v)]^3|, \mu(u, v) < \sigma(u) \wedge \sigma(v)$$

Step 4. Arrange the n^2distinct edge labels into an $n \times n$ matrix M.

Step 5. Normalize the values in M and order them. Assign each unique edge value to an alphabet symbol $(A, B, \ldots, Z)$.

10.48047/jocaaa.2024.33.07.127

Step 6. For each letter in the plain text message, substitute it with the corresponding mapped letter according to the FCDL matrix.

Step 7. Output the cipher text.

Encryption Program:

The **steps 1 to 4** to proceed with the MATLAB program are shown below. Here is the MATLAB code for FCDL graph generation:

```
% FCDL Graph with BOTH Vertex and Edge Labels
clc; clear; close all;

n = 5;
sigma_left  = [0.12 0.09 0.36 0.45 0.77];   % V1..V5
sigma_right = [0.04 0.69 0.94 0.80 0.43];   % V6..V10

% --- Compute FCDL edge weights ---
M = zeros(n,n);
for i = 1:n
   for j = 1:n
      M(i,j) = abs(sigma_left(i)^3 - sigma_right(j)^3);
   end
end
M = round(M,3);

disp('FCDL Matrix M:');
disp(M);

% --- Node positions (top row: left set, bottom row: right set) ---
xL = 1:n; yL = ones(1,n)*2;   % top row (V1..V5)
```

10.48047/jocaaa.2024.33.07.127

```
xR = 1:n; yR = zeros(1,n);    % bottom row (V6..V10)

figure('Color','w'); hold on;

% --- Plot left partition with σ-values ---
for i = 1:n
   plot(xL(i), yL(i), 'ko', 'MarkerSize', 10, 'MarkerFaceColor','c');
   text(xL(i), yL(i)+0.25, sprintf('V%d=%.2f', i, sigma_left(i)), ...
       'FontSize',9, 'FontWeight','bold', 'HorizontalAlignment','center');
end

% --- Plot right partition with σ-values ---
for j = 1:n
   plot(xR(j), yR(j), 'ko', 'MarkerSize', 10, 'MarkerFaceColor','m');
   text(xR(j), yR(j)-0.25, sprintf('V%d=%.2f', j+n, sigma_right(j)), ...
       'FontSize',9, 'FontWeight','bold', 'HorizontalAlignment','center');
end

% --- Draw edges with μ(u,v) labels ---
for i = 1:n
   for j = 1:n
      % Edge line
      plot([xL(i), xR(j)], [yL(i), yR(j)], 'k-');

      % Position of label (spread placement like reference)
      if mod(i+j,2)==0
         % Closer to left
         xm = xL(i) + 0.3 * (xR(j)-xL(i));
         ym = yL(i) + 0.3 * (yR(j)-yL(i));
      else
         % Closer to right
```

10.48047/jocaaa.2024.33.07.127

```
        xm = xL(i) + 0.7 * (xR(j)-xL(i));
        ym = yL(i) + 0.7 * (yR(j)-yL(i));
    end

    % Edge label
    text(xm, ym, sprintf('%.3f', M(i,j)), ...
        'FontSize',8, 'Color','r', 'FontWeight','bold', ...
        'HorizontalAlignment','center');
  end
end

title('FCDL Labeling of K_{5,5}', 'FontSize',12);
axis off; axis equal; hold off;
```

PYTHON PROGRAM:

Now, we proceed with python to execute remaining steps, as shown in the following program below:

```
import numpy as np

# --- Step 1: Load the CSV file ---

file_path = r"C:\Users\SONY\Desktop\FCDL_matrix.csv"

M = np.loadtxt(file_path, delimiter=",")

print("FCDL Matrix:\n", M)

# --- Step 2: Show Graph Edges with Weights ---

n = M.shape[0]
```

10.48047/jocaaa.2024.33.07.127

```
print("\nGraph Edges with Weights:")

for i in range(n):

    for j in range(n):

        if i != j and M[i, j] != 0:

            print(f"V{i+1} -- V{j+1} : weight = {M[i,j]}")

# --- Step 3: Generate Key Sequence (number + letter) ---

alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

key_numbers = (M.flatten() * 100 % 26).astype(int)

key_letters = [alphabet[num] for num in key_numbers]

key_sequence = [f"{num}({letter})" for num, letter in zip(key_numbers, key_letters)]

print("\nKey sequence (number(letter)):")

print(", ".join(key_sequence))

# --- Step 4: Encryption and Decryption ---

def encrypt(message, key_numbers):

    message = message.replace(" ", "")  # ignore spaces

    encrypted = ""

    for i, ch in enumerate(message.upper()):

        if ch in alphabet:
```

10.48047/jocaaa.2024.33.07.127

```
            shift = key_numbers[i % len(key_numbers)]
            new_index = (alphabet.index(ch) + shift) % 26
            encrypted += alphabet[new_index]
        else:
            encrypted += ch
    return encrypted
def decrypt(ciphertext, key_numbers):
    decrypted = ""
    for i, ch in enumerate(ciphertext.upper()):
        if ch in alphabet:
            shift = key_numbers[i % len(key_numbers)]
            new_index = (alphabet.index(ch) - shift) % 26
            decrypted += alphabet[new_index]
        else:
            decrypted += ch
    return decrypted
# --- Step 5: Example Encryption ---
plaintext = "FUZZY CUBE DIFFERENCE LABELING"
```

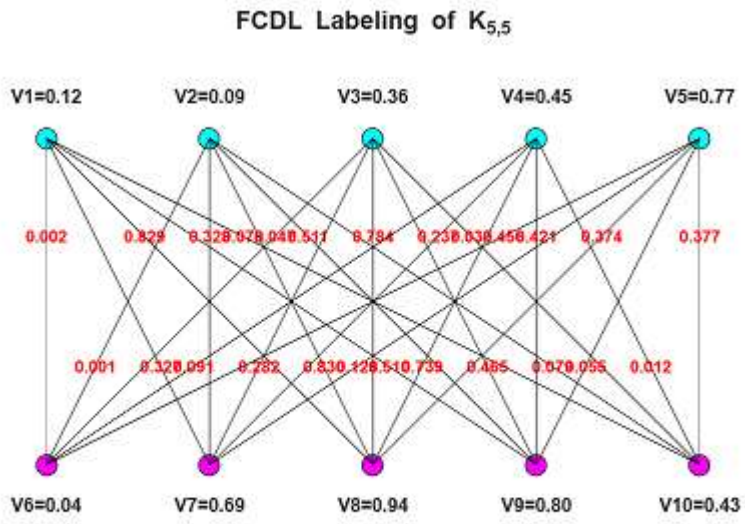
10.48047/jocaaa.2024.33.07.127

```
ciphertext = encrypt(plaintext, key_numbers)

decrypted = decrypt(ciphertext, key_numbers)

print("\nPlaintext :", plaintext)

print("Ciphertext:", ciphertext)

print("Decrypted :", decrypted)
```

Output (For MATLAB)

```
FCDL Matrix M:

    0.0020    0.3270    0.8290    0.5100    0.0780

    0.0010    0.3280    0.8300    0.5110    0.0790

    0.0470    0.2820    0.7840    0.4650    0.0330

    0.0910    0.2370    0.7390    0.4210    0.0120

    0.4560    0.1280    0.3740    0.0550    0.3770
```

10.48047/jocaaa.2024.33.07.127



Output (For Python)

FCDL Matrix:

[[0.002 0.327 0.829 0.51 0.078]

[0.001 0.328 0.83 0.511 0.079]

[0.047 0.282 0.784 0.465 0.033]

[0.091 0.237 0.739 0.421 0.012]

[0.456 0.128 0.374 0.055 0.377]]

Graph Edges with Weights:

V1 -- V2: weight = 0.327

V1 -- V3: weight = 0.829

V1 -- V4: weight = 0.51

10.48047/jocaaa.2024.33.07.127

V1 -- V5: weight = 0.078

V2 -- V1: weight = 0.001

V2 -- V3: weight = 0.83

V2 -- V4: weight = 0.511

V2 -- V5: weight = 0.079

V3 -- V1: weight = 0.047

V3 -- V2: weight = 0.282

V3 -- V4: weight = 0.465

V3 -- V5: weight = 0.033

V4 -- V1: weight = 0.091

V4 -- V2: weight = 0.237

V4 -- V3: weight = 0.739

V4 -- V5: weight = 0.012

V5 -- V1: weight = 0.456

V5 -- V2: weight = 0.128

V5 -- V3: weight = 0.374

V5 -- V4: weight = 0.055

Key sequence (number (letter)):

10.48047/jocaaa.2024.33.07.127

0(A), 6(G), 4(E), 25(Z), 7(H), 0(A), 6(G), 5(F), 25(Z), 7(H), 4(E), 2(C), 0(A), 20(U), 3(D), 9(J), 23(X), 21(V), 16(Q), 1(B), 19(T), 12(M), 11(L), 5(F), 11(L)

Plaintext: FUZZY CUBE DIFFERENCE LABELING

Ciphertext: FADYFCAGDKMHFYUNKXUMTNPQTNM

3.0.2 Decryption Algorithm (FCDL-based)

Input: Cipher text message, integer n.

Output: Plain text message.

Step 1. From the shared key (or from n), reconstruct the $n \times n$ FCDL matrix.

Step 2. Compute the same vertex labels $\sigma(v)$ and induced edge labels $\mu(u, v)$ as in encryption.

Step 3. Regenerate the substitution mapping between the alphabet and the cube difference values.

Step 4. For each letter in the cipher text, substitute it with its original letter using the inverse mapping.

Step 5. Output the plain text.

DECRYPTION PROGRAM:

```
import numpy as np

# FCDL Matrix (shared key from encryption side)

M = np.array([

   [0.002, 0.327, 0.829, 0.510, 0.078],
```

10.48047/jocaaa.2024.33.07.127

```
    [0.001, 0.328, 0.830, 0.511, 0.079],
    [0.047, 0.282, 0.784, 0.465, 0.033],
    [0.091, 0.237, 0.739, 0.421, 0.012],
    [0.456, 0.128, 0.374, 0.055, 0.377]
])
alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
# Regenerate substitution mapping (key numbers)
key_numbers = (M.flatten() * 100 % 26).astype(int)
def decrypt(ciphertext, key_numbers):
    decrypted = ""
    for i, ch in enumerate(ciphertext.upper()):
        if ch in alphabet:
            shift = key_numbers[i % len(key_numbers)]
            new_index = (alphabet.index(ch) - shift) % 26
            decrypted += alphabet[new_index]
        else:
            decrypted += ch
    return decrypted
```

10.48047/jocaaa.2024.33.07.127

```
# Example ciphertext

ciphertext = "FADYFCAGDKMHFYUNKXUMTNPQTNM"

plaintext = decrypt(ciphertext, key_numbers)

print("Ciphertext:", ciphertext)

print("Decrypted Plaintext:", plaintext)
```

Program Output:

Ciphertext: FADYFCAGDKMHFYUNKXUMTNPQTNM

Decrypted Plaintext: FUZZYCUBEDIFFERENCELABELING

4. Conclusion

This study demonstrates Fuzzy Cube Difference Labeling (FCDL) as an effective cryptographic technique that enhances security by incorporating fuzzy vertex values and cube difference edge labels to increase entropy and resist attacks. Implementation in MATLAB and Python validates its practicality. The study is limited by its focus on small graph sizes and fixed alphabet map pings. Future work will aim to scale the approach to larger, more complex graphs, assess quantum resilience, and integrate with established cryptographic frameworks to advance graph-based post-quantum cryptography.

Conflict of interest

The authors proudly affirm that they have no conflicts of interest to declare concerning the publication of this paper

References

[1] AlEtaiwi, W. M., “Encryption algorithm using graph theory,” Journal of Scientific Research and Reports, 2014, pp. 2519–2527.

10.48047/jocaaa.2024.33.07.127

[2] Bharathi, T., "Fuzzy Square Difference Labeling," Journal of Computational Mathematica, 2021, 5(1), pp. 70–75. ISSN: 2456–8686. https://doi.org/10.26524/cm93

[3] Gallian, J. A., "A dynamic survey of graph labeling," The Electronic Journal of Combina torics, Vol. 17, 2014.

[4] Gani, A., and Subahashini, M., "Properties of Fuzzy Labeling Graph," Applied Mathematical Sciences, 2012, 6(70), 3461–3466.

[5] Gani, A., Akram, M., and Subahashini, M., "Novel Properties of Fuzzy Labeling Graphs," Journal of Mathematics, 2014, pp. 1–6. https://doi.org/10.1155/2014/375135

[6] Giridaran, M., "Application of Super Magic Labeling in Cryptography," International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), 2020, 9(6), 2320 6710. https://doi.org/10.15680/IJIRSET.2020.0906003

[7] Katz, J., and Lindell, Y., Introduction to Modern Cryptography, CRC Press, 2014.

[8] Rosa, A., "On certain valuations of the vertices of a graph," in Theory of Graphs (International Symposium, Rome), 1966, pp. 349–355.

[9] Shiama, J., "Cube Difference Labeling of Some Graphs," IJESIT, 2013, 2(6), 2319–5967. [10] Yamuna, M., Suwathi, A., and Krishnan, N., "Four Digit Pin Number as a Digraph," Inter national Journal of Computer Application, Vol. 3, Issue 4, 2014, pp. 100–107. [11] Zadeh, L. A., "Fuzzy sets," Information and Control, 1965, 8(3), 338–353. https://doi.org/ 10.2307/2272014