World Journal of Advanced Research and Reviews
eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(RESEARCH ARTICLE)

Check for updates

An enhancement of four-square with zigzag transformation encryption algorithm based on 3d rubik's cube principle

Anne Lorrea Almerol *, Jose Cyril Rupinta, Jamel Susi, Raymund Dioses, Elsa Pascual, Florencio Contreras Jr., Jamillah Guialil and Jonathan Morano

Department of Computer Science, College of Information Systems and Technology Management, Pamantasan ng Lungsod ng Maynila, Manila, Philippines.

World Journal of Advanced Research and Reviews, 2024, 22(01), 1339–1359

Publication history: Received on 02 March 2024; revised on 17 April 2024; accepted on 20 April 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.22.1.1135

Abstract

The existing Four-Square cipher, specifically the Four-Square with Zigzag transformation encryption algorithm, serves as the foundation of this study, aiming to solve its cryptographic limitations. The existing algorithm cannot encrypt messages with numbers and special characters, the keys can be easily cracked, and when the process is repeated more than 26 times, the encrypted digraph is the same as the first encrypted digraph. This study aims to enhance the existing algorithm by transforming the 5x5 matrix, enhancing the encryption-decryption key, and improving the Zigzag transformation. The methodology employed involves utilizing a 6x6x6 cube to include uppercase and lowercase letters, numbers, and special characters. Random encryption-decryption keys are generated using the Cryptographically Secure Pseudorandom Number Generator (CSPRNG), Fibonacci sequence, Tribonacci sequence, and Linear Feedback Shift Register. Zigzag transformation is improved by employing Rubik's cube principle, CSPRNG, Fibonacci sequence, and Tribonacci sequence to randomize the cube rotation. Various tests were conducted to evaluate the enhanced algorithm. The matrix comparison test demonstrated a significant expansion in the character set, allowing the utilization of uppercase and lowercase letters, numbers, and special characters. The comparison of encrypted and decrypted text highlighted the enhanced algorithm's ability to revert ciphertext into the original plaintext, surpassing the limitations of the existing algorithm. Statistical randomness tests, including the Frequency (Monobit) and Runs tests, provided robust evidence of the randomness of the algorithm, meeting the threshold for secure encryption. The average avalanche effect of the enhanced algorithm is 52.78%, surpassing the minimum avalanche effect of a secure cryptographic algorithm.

Keywords: Four-square; Cryptography; Encryption; Zigzag transformation; Key

1. Introduction

Cryptography is the study and application of techniques for securing communication and protecting information from unauthorized access or tampering. To enable secure communication in the context of possible adversaries is the main objective of cryptography. It entails employing encryption and decryption techniques to convert plain text—readable information—into cipher text or unreadable information, and vice versa, using mathematical procedures. As technology advances, there is a demand for novel and secure cryptographic methods.

In 1898, the French cryptographer Felix Delastelle introduced the four-square cipher, a manual symmetric encryption method [1]. The four 5x5 matrices stacked in a square form the basis of the technique. Except for one letter—typically the letter "Q" or another less common letter, each matrix has a distinct set of the alphabet. Letter pairs from the plaintext are separated and placed on the key matrices. These pairs are converted into ciphertext according to rules involving the

* Corresponding author: Anne Lorrea Almerol

matrices' rows and columns. The symmetry of the procedure adds to the cipher's elegance by guaranteeing that the decryption is as simple as the encryption [1]. The Four-Square cipher is still vulnerable to contemporary cryptographic analysis despite its significant advancement in cryptographic design at its creation. The Four-Square cipher encrypts pairs of letters (like Playfair), making it significantly more robust than substitution ciphers; however, it can be easily cracked if plaintext and ciphertext are known [2]. Numerous researchers have already worked on more popular encryption techniques, such as Playfair, Polybius, and Hill Cipher. However, expanded versions of Four-Square ciphers are uncommon, so research on an extended version of Four-Square ciphers that is still very simple and adaptable to implement is needed.

The 5x5 Polybius square is vulnerable to cryptographic weaknesses due to several constraints despite its historical significance. The 5x5 Polybius square's limited key space is its primary weakness. Because there are only 25 spaces in the grid—which may hold all 26 letters of the alphabet, except for the letter "J"—brute-force attacks can exploit the encryption [3, 4]. Due to its limitations, the 5x5 Polybius square is no longer relevant in contemporary security standards, regardless of its historical relevance as an early attempt at cryptographic encoding. Its limited key space, vulnerability to known-plaintext attacks, susceptibility to frequency analysis, lack of confusion, and weakness against sophisticated cryptanalysis techniques underscore the significance of adopting more resilient and secure encryption mechanisms to protect sensitive data in the modern digital age [5].

Elazzaby, El Akkad, and Kabbaj [6] enhanced the algorithm of Four-Square cipher by developing a new encryption approach based on Four-Square ciphers and Zigzag transformation to resist cryptanalysis attacks such as brute attacks and statistic attacks, slowing down the decryption process by having different encryption in each digraph in a message. However, this algorithm has limitations: (1) It cannot encrypt messages with numbers and special characters due to its limitations in using a 5x5 Polybius square matrix, (2) The keys can be easily cracked if both the plaintext and ciphertext are known, and (3) when the process is repeated to the same digraph more than 26 times, the encrypted digraph is the same as the first encrypted digraph.

This study aims to enhance the existing Four-Square with Zigzag transformation encryption by adding necessary security and complexity when encrypting texts. Specifically, this study is intended to: (1) Transform the 5x5 matrix into a 6x6x6 cube containing a total of 216 characters, which includes uppercase letters, lowercase letters, numbers, and special characters, (2) Enhance the keys by applying Cryptographically Secure Pseudorandom Number Generator (CSPRNG), Fibonacci sequence, Tribonacci sequence and Linear Feedback Shift Register (LFSR) to randomize the encryption-decryption keys, and (3) Enhance the Zigzag transformation by applying the CSPRNG, Fibonacci sequence, and Tribonacci sequence to randomize the number of rotations of the cube.

2. Material and methods

2.1. Materials

In this study, the Python programming language is used to develop the simulator of the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle. PyCharm and Visual Studio Code IDE are used to implement the Python code. The Python modules used are NumPy, secrets and LFSR from the pylfsr package. The characters used in the encryption and decryption process are based on the standard ASCII printable characters and extended ASCII printable characters from ASCII CP437 [7].

2.2. Notations and Descriptions

The table below shows the notations used in this paper to discuss the methods of the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle.

Table 1 Notation and Descriptions

Notation	Description	Notation	Description
PT	The plaintext to be encrypted.	CT	The ciphertext to be decrypted.
D	The digraph of the plaintext or ciphertext	F1	The front face of the cube
x	The position of the digraph	F2	The right face of the cube
$D1_x$	The first character of the digraph based on position x	F3	The back face of the cube

$D2_x$	The second character of the digraph based on position x	F4	The left face of the cube
l	The half of the length of the plaintext or ciphertext	F5	The top face of the cube
UL	One of the plaintext cubes. The upper-left cube	F6	The bottom face of the cube
UR	One of the ciphertext cubes. The upper-right cube	$D\text{-}D1_x$	The decrypted character of the D1 based on position x
LL	One of the ciphertext cubes. The lower-left cube	$D\text{-}D2_x$	The decrypted character of the D2 based on position x
LR	One of the plaintext cubes. The lower-right cube	n	The first cryptographically secure random number generated
K1	The first encryption or decryption key	m	The second cryptographically secure random number generated
K2	The second encryption or decryption key	FIB_n	The Fibonacci number based on position n
E-K1	The first random encrypted key	TRI_m	The Tribonacci number based on position m
E-K2	The second random encrypted key	R_{UR}	The row number of UR to be rotated
FN_{CUBE}	The face number of the character in the cube	R_{LL}	The row number of LL to be rotated
R_{CUBE}	The row number of the character in the cube	aUR_x	The angle (in degrees) that the F1 of UR would be rotated
C_{CUBE}	The column number of the character in the cube	aLL_x	The angle (in degrees) that the F1 of LL would be rotated
$E\text{-}D1_x$	The encrypted character of the D1 based on position x	RR_{UR}	The number of times R_{UR} would be rotated
$E\text{-}D2_x$	The encrypted character of the D2 based on position x	RR_{LL}	The number of times R_{LL} would be rotated

2.3. Random Number Generation

2.3.1. Cryptographically Secure Pseudorandom Number Generator (CSPRNG)

Two cryptographically secure pseudorandom numbers are generated using the secrets module in Python. The range of cryptographically secure pseudorandom numbers is from 1 to 216. The first cryptographically secure pseudorandom number generated will be used to determine the position of the random first Fibonacci sequence number, and the second cryptographically secure pseudorandom number generated will be used to determine the position of the random first Tribonacci sequence number.

2.3.2. Fibonacci Sequence

In this study, the Fibonacci sequence is used to randomize the first key, the number of times a row of the upper-right cube would be rotated, and the angle (in degrees) of the front face of the upper-right cube would be rotated. Only the first 216 Fibonacci numbers will be used in this study. Once the 216^{th} Fibonacci number is reached, the next Fibonacci number will be the first Fibonacci number, followed by the second Fibonacci number, and so forth.

The initial position of the random first Fibonacci sequence number is determined by using the first cryptographically secure pseudorandom number generated in the previous step. The last number of the Fibonacci sequence number depends on the length of the key or the plaintext.

2.3.3. Tribonacci Sequence

In this study, the Tribonacci sequence is used to randomize the second key, the number of times a row of the lower-left cube would be rotated, and the angle (in degrees) of the front face of the lower-left cube would be rotated. Only the first 216 Tribonacci numbers will be used in this study. Once the 216th Tribonacci number is reached, the next Tribonacci number will be the first Tribonacci number, followed by the second Tribonacci number, and so forth.

The initial position of the first random Tribonacci sequence number is determined using the second cryptographically secure pseudorandom number generated in the previous step. The last number of the random Tribonacci sequence number depends on the length of the key or plaintext.

2.4. Random Key Generation

The Linear Feedback Shift Register (LFSR) model is used to generate a random key. The pylfsr package in Python programming language is used to create the Linear Feedback Shift Register (LFSR) model. The model is set to an 8-bit LFSR model. The taps of the LFSR model should create an output with a maximum length [8, 9]. The taps are set to 8, 6, 5, and 4 to generate an output with a maximum length.

The two initial keys, random Fibonacci sequence numbers and random Tribonacci sequence numbers are used to generate the random initial seed of the LFSR model. The remainder when the Fibonacci sequence number is divided by 256 and the remainder when the Tribonacci sequence number is divided by 256 are computed to get the first and second random numbers, respectively. The two initial keys, the first and second random numbers, are converted into their equivalent binary form. The XOR operation between the first initial key and the first random number determines the initial seed of the first LFSR model. The XOR operation between the second initial key and the second random number determines the initial seed of the second LFSR model.

Figure 1 shows the process of the LFSR model, where the rightmost bit from the shift register pops out and is reused to XORed taps, which push the output into the leftmost bit, shifting every bit to the right. The process is repeated until the model processes all the characters of the key [9]. The binary output of the LFSR is converted to its equivalent CP437 ASCII character. If the character is not included in the first 216 printable characters, the bits will be shifted to the left until the character is included in the first 216 printable characters. The process will be repeated for each character of the two initial keys to generate two random (encrypted) keys. The two random encrypted keys will be used as the key in the ciphertext matrices.

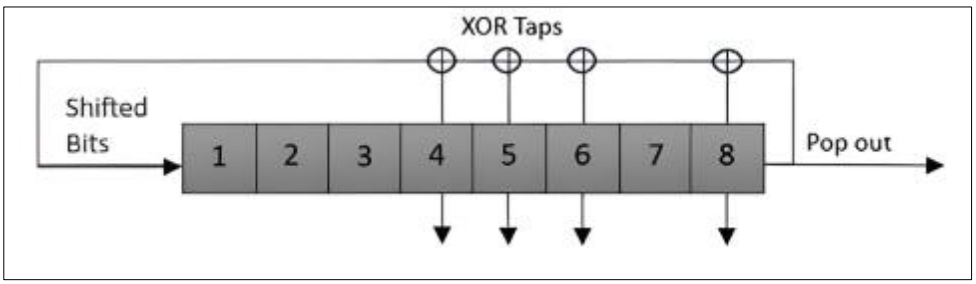

Figure 1 8-bit LFSR with Feedback Polynomial x8+x6+ x5 + x4 with Maximum Length of 255

2.5. Four 6x6x6 Cubes

Four 6x6x6 cubes are used to generate the two plaintext and ciphertext cubes. Each cube contains 216 characters from the standard ASCII printable and ASCII CP437 extended characters [7]. The arrangement of characters of the ciphertext cubes depends on the two random encrypted keys. The first random encrypted key is used in the upper-right cube. The second random encrypted key is used in the lower-left cube. The ciphertext cube will be filled with the unique characters of random encrypted keys, followed by the remaining characters that do not appear in the random encrypted keys. The remaining characters are arranged based on the ASCII table [7]. The sequence in filling the squares of the faces of the cubes is front face, right face, back face, left face, top face, and bottom face. The four 6x6x6 cube in the expanded version is shown in Figure 2.

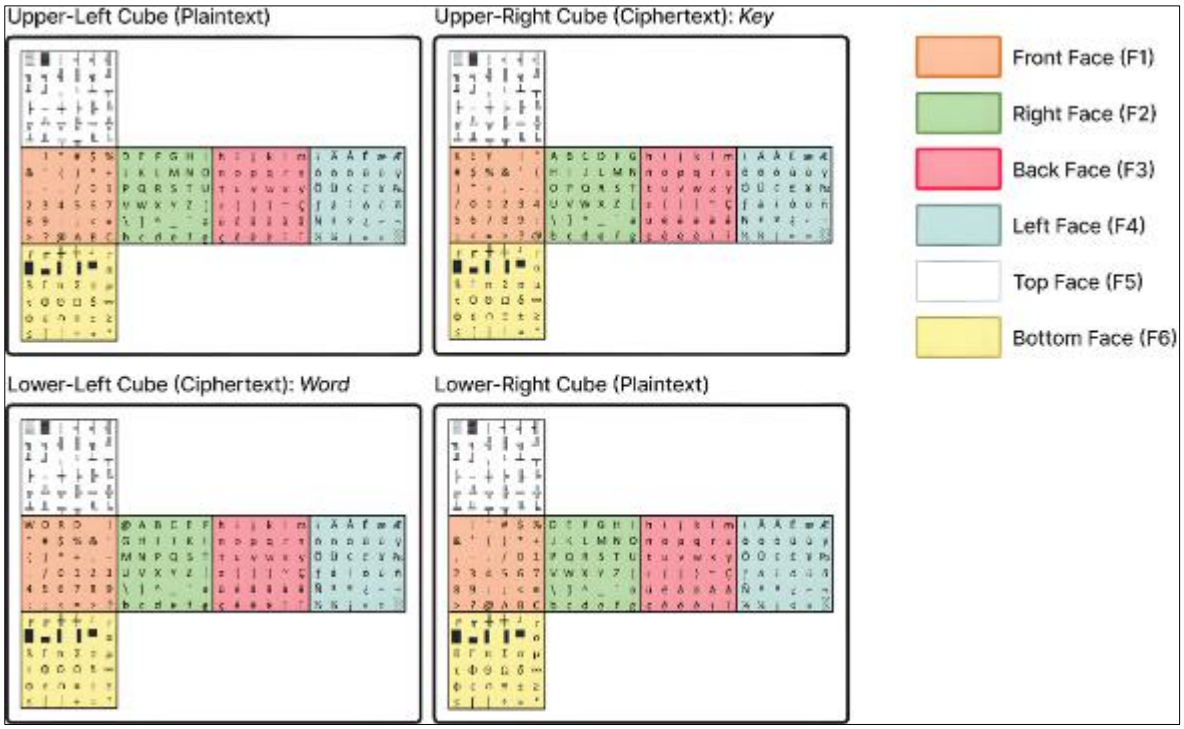

Figure 2 216 ASCII Printable and Extended Printable Characters (Expanded Version of Four 6x6x6 Cube)

2.6. Enhanced Zigzag Transformation

Cryptographically Secure Pseudorandom Number Generator (CSPRNG), Fibonacci sequence numbers, and Tribonacci sequence numbers are used in the Zigzag transformation process to randomly rotate the row and the angle of the front face of the two ciphertext cubes.

The Fibonacci sequence, Tribonacci sequence, Modulus operation, and Addition operation randomly rotate the row and angle of the front face of the upper-right and lower-left cube in the encryption and decryption process. The rotation of the row of the upper-right and lower-left cube is to the right and the left, respectively. The number of times the row can be rotated is one, two, and three times. The degrees the front face of the cube can be rotated are 0 degree, 90 degrees, 180 degrees, and 270 degrees.

2.6.1. Upper-Right Cube Rotation

The random angle of rotation of the front face of the upper-right cube for each digraph is computed by getting the remainder when the corresponding Fibonacci number based on the position of the current digraph to be encrypted or decrypted is divided by 4 and multiplied by 90. This can be expressed using the Equation (1):

$$aUR_x = (FIB_n \ mod \ 4) \ * \ 90 \text{.....(1)}$$

where n is the position of the Fibonacci number from the random Fibonacci sequence generated from the previous step, aUR_x is the upper-right cube front face angle rotation for the digraph x to be encrypted or decrypted, FIB_n is the Fibonacci number obtained from the random Fibonacci sequence generated in the previous step.

The random number of rotations of the row of the upper-right cube for each digraph is computed by getting the remainder when the corresponding Fibonacci number based on the position of the current digraph to be encrypted or decrypted is divided by 3 and added by 1. This can be expressed using the Equation (2):

$$RR_{UR} = (FIB_n \ mod \ 3) \ + 1 \text{.....(2)}$$

where n is the position of the Fibonacci number from the random Fibonacci sequence generated from the previous step, RR_{UR} is the upper-right cube row rotation for the current digraph to be encrypted or decrypted, FIB_n is the Fibonacci number obtained from the random Fibonacci sequence generated in the previous step.

2.6.2. Lower-Left Cube Rotation

The random angle of rotation of the front face of the lower-left cube for each digraph is computed by getting the remainder when the corresponding Tribonacci number based on the position of the current digraph to be encrypted or decrypted is divided by 4 and multiplied by 90. This can be expressed using the Equation (3):

$$aLL_x = (TRI_m \ mod \ 4) \ * \ 90 \text{.....(3)}$$

where m is the position of the Tribonacci number from the random Tribonacci sequence generated from the previous step, aLL_x is the lower-left cube front face angle rotation for the digraph x to be encrypted or decrypted, TRI_m is the Tribonacci number obtained from the random Tribonacci sequence generated in the previous step.

The random number of rotations of the row of the lower-left cube for each digraph is computed by getting the remainder when the corresponding Tribonacci number based on the position of the current digraph to be encrypted or decrypted is divided by 3 and added by 1. This can be expressed using the Equation (4):

$$RR_{LL} = (TRI_m \ mod \ 3) \ + \ 1 \text{.....(4)}$$

where m is the position of the Tribonacci number from the random Tribonacci sequence generated from the previous step, RR_{LL} is the lower-left cube row rotation for the current digraph to be encrypted or decrypted, TRI_m is the Tribonacci number obtained from the random Tribonacci sequence generated in the previous step.

2.7. Conceptual Framework

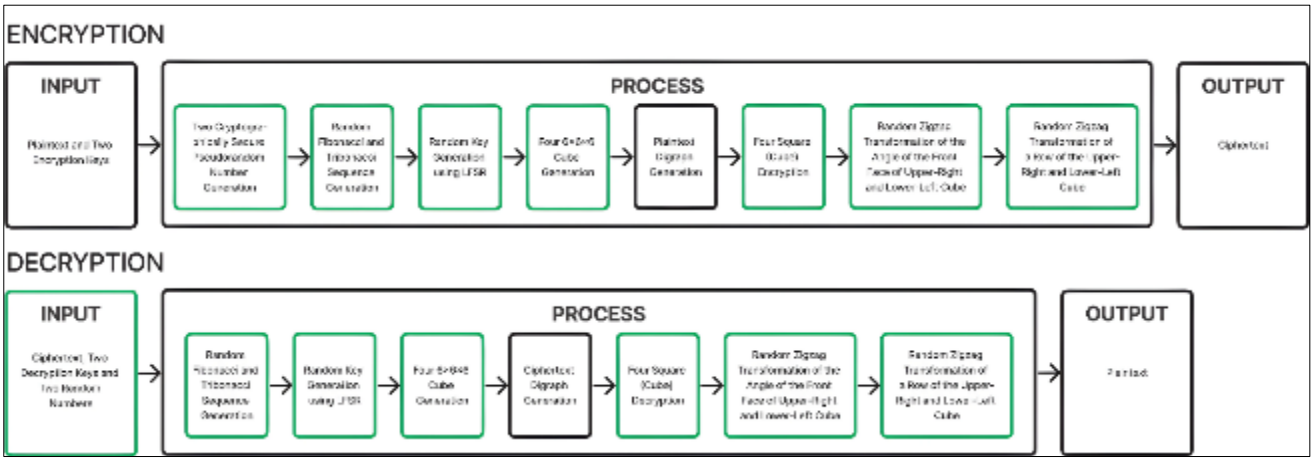

Figure 3 Conceptual Framework of Encryption and Decryption Process of the Enhanced Four-Square with Zigzag Transformation Encryption Algorithm based on 3D Rubik's Cube Principle

2.8. Pseudocode

2.8.1. Encryption Process

1. Input PT, K1 and K2.
2. Generate n and m.
3. Get FIB_n and TRI_m.
4. Generate random E-K1 and E-K2 using LFSR.
   4.1. Generate two 8-bit LFSR (LFSR1 and LFSR2).
   4.2. LFSR1 and LFSR2 taps = 8, 6, 5, 4
   4.3. randFib = FIB_n mod 256 and randTri = TRI_m mod 256
   4.4. For each character in K1:
      4.4.1. charK1 = convertToBinary(character)
      4.4.2. binaryFIB = convertToBinary(randFib)
      4.4.3. LFSR initial state = charK1 ⊕ binaryFIB
      4.4.4. LFSR-K1 = output of LFSR1
      4.4.5. binK1 = convertToString(LFSR-K1)
      4.4.6. While binK1 is not included in the first 216 printable and extended printable characters:

4.4.6.1. LFSR-K1 = LFSR-K1 << 1
4.4.6.2. binK1 = convertToString(LFSR-K1)
4.4.7. randFib = (FIB_{n+1}) mod 256
4.4.8. E-K1 = E-K1 + binK1
4.5. For each character in K2:
4.5.1. charK2 = convertToBinary(character)
4.5.2. binaryTRI = convertToBinary(randTri)
4.5.3. LFSR2 initial state = charK2 ⊕ binaryTRI
4.5.4. LFSR-K2 = output of LFSR2
4.5.5. binK2 = convertToString(LFSR-K2)
4.5.6. While binK2 is not included in the first 216 printable and extended printable characters:
4.5.6.1. LFSR-K2 = LFSR-K2 << 1
4.5.6.2. binK2 = convertToString(LFSR-K2)
4.5.7. randTri= (TRI_{m+1}) mod 256
4.5.8. E-K2 = E-K2 + binK2

5. Generate four 6x6x6 cubes and arrange them in a square.
5.1. Fill the squares of F1, F2, F3, F4, F5, and F6 of the UL and LR with the printable and extended printable characters of the ASCII Table.
5.2. Fill the squares of F1, F2, F3, F4, F5, and F6 of the UR with the unique characters of E-K1, followed by the printable and extended printable characters of the ASCII Table that do not appear in E-K1.
5.3. Fill the squares of F1, F2, F3, F4, F5, and F6 of the LL with the unique characters of E-K2, followed by the printable and extended printable characters of the ASCII Table that do not appear in E-K2.
6. Break up the PT into D.
6.1. If the length of the PT is odd, append the letter 'X' at the end.
7. Locate the position of the current D_x to be encrypted in UL and LR.
7.1. Determine FN_{UL}, R_{UL}, and C_{UL} of the $D1_x$ in the UL.
7.2. Determine FN_{LR}, R_{LR}, and C_{LR} of the $D2_x$ in the LR.
8. Determine the character in the UR and LL around the corners of the rectangle the characters of the D_x of the PT created.
8.1. E-$D1_x$ is the character in UR at FN_{LR}, C_{LR}, and R_{UL}.
8.2. E-$D2_x$ is the character in LL at FN_{UL}, C_{UL}, and R_{LR}.
8.3. Append E-$D1_x$ and E-$D2_x$ to the CT.
8.4. Set $F1_{UR} = FN_{LR}$, $F1_{LL} = FN_{UL}$, $R_{UR} = R_{UL}$, and $R_{LL} = R_{LR}$.
9. Apply zigzag transformation to the UR and LL.
9.1. Rotate the angle of $F1_{UR}$: $aUR_x = (FIB_n \bmod 4) * 90$)
9.2. Rotate the R_{UR} to the right: $RR_{UR} = (FIB_n \bmod 3) + 1$
9.3. Rotate the angle of $F1_{LL}$: $aLL_x = (TRI_m \bmod 4) * 90$)
9.4. Rotate the R_{LL} to the left: $RR_{LL} = (TRI_m \bmod 3) + 1$
9.5. $FIB_n = FIB_{n+1}$ and $TRI_m = TRI_{m+1}$
10. Repeat Step 7-9 until all the D of the PT are encrypted.

2.8.2. Decryption Process

1. Input CT, K1, K2, n and m.
2. Get FIB_n and TRI_m.
3. Generate E-K1 and E-K2 using LFSR.
3.1. Generate two 8-bit LFSR (LFSR1 and LFSR2).
3.2. LFSR1 and LFSR2 taps = 8, 6, 5, 4
3.3. randFib = FIB_n mod 256 and randTri = TRI_m mod 256
3.4. For each character in K1:
3.4.1. charK1 = convertToBinary(character)
3.4.2. binaryFIB = convertToBinary(randFib)
3.4.3. LFSR initial state = charK1 ⊕ binaryFIB
3.4.4. LFSR-K1 = output of LFSR1
3.4.5. binK1 = convertToString(LFSR-K1)
3.4.6. While binK1 is not included in the first 216 printable and extended printable characters:
3.4.6.1. LFSR-K1 = LFSR-K1 << 1
3.4.6.2. binK1 = convertToString(LFSR-K1)
3.4.7. randFib = (FIB_{n+1}) mod 256

3.4.8. E-K1 = E-K1 + binK1
3.5. For each character in K2:
3.5.1. charK2 = convertToBinary(character)
3.5.2. binaryTRI = convertToBinary(randTri)
3.5.3. LFSR2 initial state = charK2 ⊕ binaryTRI
3.5.4. LFSR-K2 = output of LFSR2
3.5.5. binK2 = convertToString(LFSR-K2)
3.5.6. While binK2 is not included in the first 216 printable and extended printable characters:
3.5.6.1. LFSR-K2 = LFSR-K2 << 1
3.5.6.2. binK2 = convertToString(LFSR-K2)
3.5.7. randTri= (TRI_{m+1}) mod 256
3.5.8. E-K2 = E-K2 + binK2
4. Generate four 6x6x6 cubes and arrange them in a square.
4.1. Fill the squares of F1, F2, F3, F4, F5, and F6 of the UL and LR with the printable and extended printable characters of the ASCII Table.
4.2. Fill the squares of F1, F2, F3, F4, F5, and F6 of the UR with the unique characters of E-K1, followed by the printable and extended printable characters of the ASCII Table that do not appear in E-K1.
4.3. Fill the squares of F1, F2, F3, F4, F5, and F6 of the LL with the unique characters of E-K2, followed by the printable and extended printable characters of the ASCII Table that do not appear in E-K2.
5. Break up the CT into D.
6. Locate the position of the current D_x to be decrypted in UR and LL
6.1. Determine FN_{UR}, R_{UR}, and C_{UR} of the $D1_x$ in the UR.
6.2. Determine FN_{LL}, R_{LL}, and C_{LL} of the $D2_x$ in the LL.
7. Determine the character in the UL and LR around the corners of the rectangle the characters of the D_x of the CT created.
7.1. D-$D1_x$ is the character in UL at FN_{LL}, C_{LL}, and R_{UR}.
7.2. D-$D2_x$ is the character in LR at FN_{UR}, C_{UR}, and R_{LL}.
7.3. Append D-$D1_x$ and D-$D2_x$ to the PT.
7.4. Set $F1_{UR} = FN_{UR}$ and $F1_{LL} = FN_{LL}$
8. Apply zigzag transformation to the UR and LL
8.1. Rotate the angle of $F1_{UR}$: $aUR_x = (FIB_n \bmod 4) * 90$)
8.2. Rotate the R_{UR} to the right: $RR_{UR} = (FIB_n \bmod 3) + 1$
8.3. Rotate the angle of $F1_{LL}$: $aLL_x = (TRI_m \bmod 4) * 90$)
8.4. Rotate the R_{LL} to the left: $RR_{LL} = (TRI_m \bmod 3) + 1$
8.5. $FIB_n = FIB_{n+1}$ and $TRI_m = TRI_{m+1}$
9. Repeat Step 6-8 until all the D of the CT are decrypted.

2.9. Simulation

2.9.1. Encryption Process

1. **Input:**
   a. Plaintext - W_4d
   b. First Key - F0^r
   c. Second Key - Cu9&
2. **Cryptographically Secure Pseudorandom Number Generation:**
   a. First Random Number – 7
   b. Second Random Number – 3
3. **Random Sequence Number Generation:**
   a. Fibonacci Sequence Numbers - 13, 21, 34, 55
   b. Tribonacci Sequence Numbers - 2, 4, 7, 13
4. **Random Key Generation:**
   a. First Random Encrypted Key
      i. First Key - F0^r
      ii. Binary Form of First Key - 01000110, 00110000, 01011110, 01110010
      iii. Fibonacci Sequence Numbers - 13, 21, 34, 55
      iv. Binary Form of Fibonacci Sequence Numbers - 00001101, 00010101, 00100010, 00110111
      v. First LFSR Initial State - 01001011, 00100101, 01111100, 01000101
      vi. First LFSR Output - 10100100, 01001000, 01111101, 01000100
      vii. First LFSR Output (Character) - ñH}D

viii. First Random Encrypted Key - ñH}D

b. Second Random Encrypted Key
   i. Second Key - Cu9&
   ii. Binary Form of Second Key - 01000011, 01110101, 00111001, 00100110
   iii. Tribonacci Sequence Numbers - 2, 4, 7, 13
   iv. Binary Form of Tribonacci Sequence Numbers - 00000010, 00000100, 00000111, 00001101
   v. Second LFSR Initial State - 01000001, 01110001, 00111110, 00101011
   vi. Second LFSR Output - 00000101, 00011100, 11111001, 10101000
   vii. Second LFSR Output (Character) - ♣∟·¿
   viii. Second Random Encrypted Key - (8≤¿

5. Plaintext Digraph Generation:
   a. length = 2
   b. First Digraph: W_
   c. Second Digraph: 4d

6. Four-Square (Cube) Encryption (Plaintext First Digraph: W_):
   a. Upper-Left Cube (Plaintext First Letter: W) - Face 2, Row 4, Column 2
   b. Lower-Right Cube (Plaintext Second Letter: _) - Face 2, Row 5, Column 4
   c. Upper-Right Cube (Ciphertext First Letter: W) - Face 2, Row 4, Column 4
   d. Lower-Left Cube (Ciphertext Second Letter: [) - Face 2, Row 5, Column 2
   e. Ciphertext - W[
   f. Upper-Right Cube Front Face (F1) - Face 2
   g. Upper-Right Cube Row Rotation: Row 4
   h. Lower-Left Cube Front Face (F1) - Face 2
   i. Lower-Left Cube Row Rotation: Row 5

7. Random Zigzag Transformation:
   a. Rotate the angle of $F1_{UR}$: $aUR_1 = (FIB_1 \bmod 4) * 90) = (13 \bmod 4) * 90 = 90$ degrees
   b. Rotate the R_{UR} to the right: $RR_{UR} = (FIB_1 \bmod 3) + 1 = (13 \bmod 3) + 1 = 2$ times
   c. Rotate the angle of $F1_{LL}$: $aLL_1 = (TRI_1 \bmod 4) * 90) = (2 \bmod 4) * 90 = 180$ degrees
   d. Rotate the R_{LL} to the left: $RR_{LL} = (TRI_1 \bmod 3) + 1 = (2 \bmod 3) + 1 = 3$ times
   e. $FIB_n = FIB_{n+1} = FIB_{1+1} = FIB_2 = 21$
   f. $TRI_m = TRI_{m+1} = TRI_{1+1} = TRI_2 = 4$

8. Four-Square (Cube) Encryption (Plaintext Second Digraph: 4d):
   a. Upper-Left Cube (Plaintext First Letter: 4) - Face 1, Row 4, Column 3
   b. Lower-Right Cube (Plaintext Second Letter: d) - Face 2, Row 6, Column 3
   c. Upper-Right Cube (Ciphertext First Letter: ┼) - Face 2, Row 4, Column 3
   d. Lower-Left Cube (Ciphertext Second Letter: E) - Face 1, Row 6, Column 3
   e. Ciphertext - W[┼E
   f. Upper-Right Cube Front Face (F1) - Face 2
   g. Upper-Right Cube Row Rotation: Row 4
   h. Lower-Left Cube Front Face (F1) - Face 1
   i. Lower-Left Cube Row Rotation: Row 6

9. Random Zigzag Transformation:
   a. Rotate the angle of $F1_{UR}$: $aUR_1 = (FIB_2 \bmod 4) * 90) = (21 \bmod 4) * 90 = 90$ degrees
   b. Rotate the R_{UR} to the right: $RR_{UR} = (FIB_2 \bmod 3) + 1 = (21 \bmod 3) + 1 = 1$ time
   c. Rotate the angle of $F1_{LL}$: $aLL_1 = (TRI_2 \bmod 4) * 90) = (4 \bmod 4) * 90 = 0$ degree
   d. Rotate the R_{LL} to the left: $RR_{LL} = (TRI_2 \bmod 3) + 1 = (4 \bmod 3) + 1 = 2$ times
   e. $FIB_n = FIB_{n+1} = FIB_{2+1} = FIB_3 = 34$
   f. $TRI_m = TRI_{m+1} = TRI_{2+1} = TRI_3 = 7$

10. Ciphertext: W[┼E

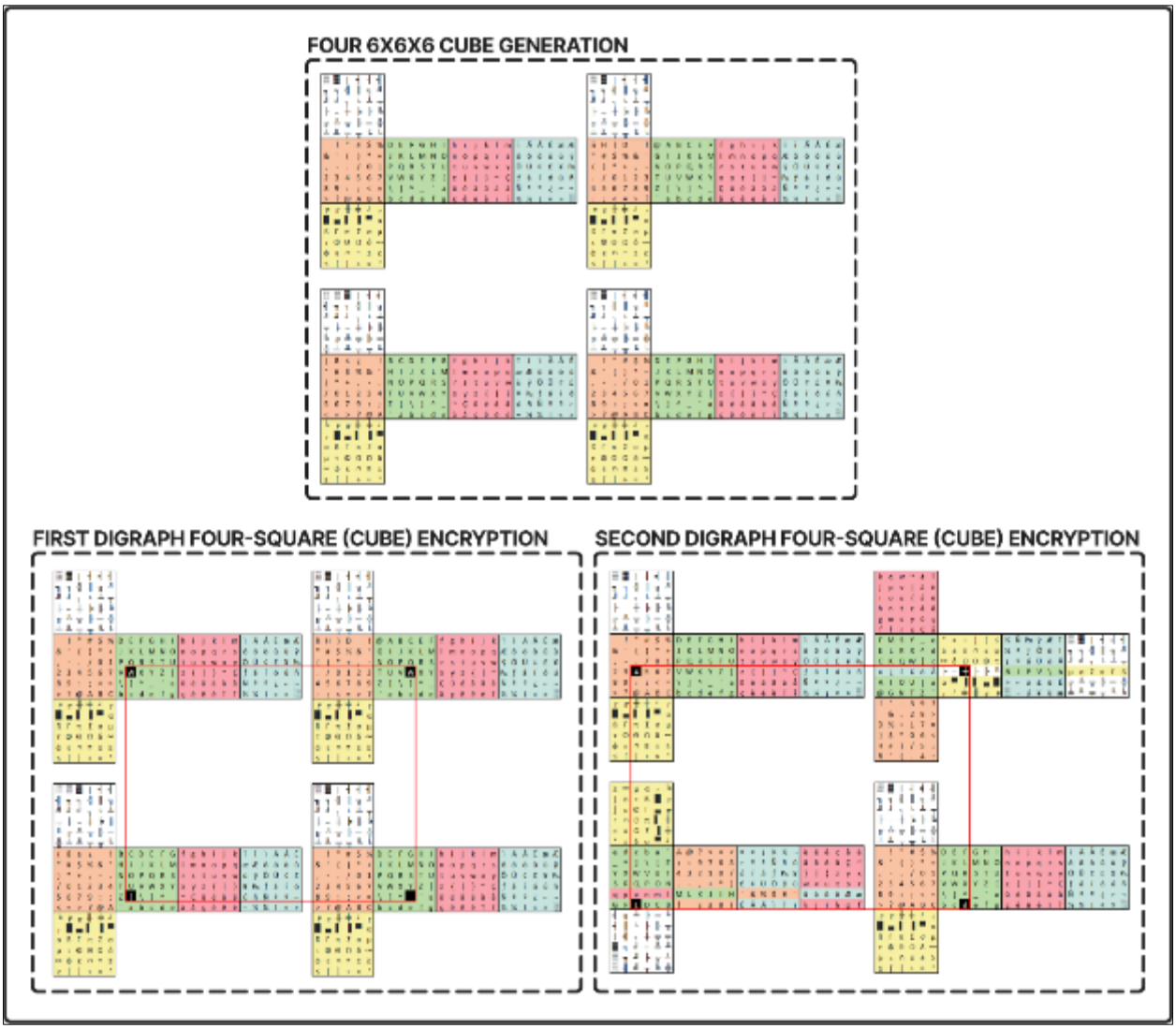

Figure 4 Encryption Process of Enhanced Four-Square with Zigzag Transformation Encryption Algorithm based on 3D Rubik's Cube Principle

2.9.2. Decryption Process

1. **Input:**
    a. Plaintext - W[┼E
    b. First Key - F0^r
    c. Second Key - Cu9&
    d. First Random Number – 7
    e. Second Random Number – 3
2. **Random Sequence Number Generation:**
    a. Fibonacci Sequence Numbers - 13, 21, 34, 55
    b. Tribonacci Sequence Numbers - 2, 4, 7, 13
3. **Random Key Generation:**
    a. First Random Encrypted Key
        i. First Key - F0^r
        ii. Binary Form of First Key - 01000110, 00110000, 01011110, 01110010
        iii. Fibonacci Sequence Numbers - 13, 21, 34, 55
        iv. Binary Form of Fibonacci Sequence Numbers - 00001101, 00010101, 00100010, 00110111
        v. First LFSR Initial State - 01001011, 00100101, 01111100, 01000101
        vi. First LFSR Output - 10100100, 01001000, 01111101, 01000100
        vii. First LFSR Output (Character) - ñH}D

viii. First Random Encrypted Key - ñH}D

b. Second Random Encrypted Key
   i. Second Key - Cu9&
   ii. Binary Form of Second Key - 01000011, 01110101, 00111001, 00100110
   iii. Tribonacci Sequence Numbers - 2, 4, 7, 13
   iv. Binary Form of Tribonacci Sequence Numbers - 00000010, 00000100, 00000111, 00001101
   v. Second LFSR Initial State - 01000001, 01110001, 00111110, 00101011
   vi. Second LFSR Output - 00000101, 00011100, 11111001, 10101000
   vii. Second LFSR Output (Character) - ♣∟·¿
   viii. Second Random Encrypted Key - (8≤¿

4. Ciphertext Digraph Generation:
   a. length = 2
   b. First Digraph: W[
   c. Second Digraph: ┼E

5. Four-Square (Cube) Decryption (Ciphertext First Digraph: W[):
   a. Upper-Right Cube (Ciphertext First Letter: W) - Face 2, Row 4, Column 4
   b. Lower-Left Cube (Ciphertext Second Letter: [) - Face 2, Row 5, Column 2
   c. Upper-Left Cube (Plaintext First Letter: W) - Face 2, Row 4, Column 2
   d. Lower-Right Cube (Plaintext Second Letter: _) - Face 2, Row 5, Column 4
   e. Plaintext – W_
   f. Upper-Right Cube Front Face (F1) - Face 2
   g. Upper-Right Cube Row Rotation: Row 4
   h. Lower-Left Cube Front Face (F1) - Face 2
   i. Lower-Left Cube Row Rotation: Row 5

6. Random Zigzag Transformation:
   a. Rotate the angle of $F1_{UR}$: $aUR_1 = (FIB_1 \bmod 4) * 90) = (13 \bmod 4) * 90 = 90$ degrees
   b. Rotate the R_{UR} to the right: $RR_{UR} = (FIB_1 \bmod 3) + 1 = (13 \bmod 3) + 1 = 2$ times
   c. Rotate the angle of $F1_{LL}$: $aLL_1 = (TRI_1 \bmod 4) * 90) = (2 \bmod 4) * 90 = 180$ degrees
   d. Rotate the R_{LL} to the left: $RR_{LL} = (TRI_1 \bmod 3) + 1 = (2 \bmod 3) + 1 = 3$ times
   e. $FIB_n = FIB_{n+1} = FIB_{1+1} = FIB_2 = 21$
   f. $TRI_m = TRI_{m+1} = TRI_{1+1} = TRI_2 = 4$

7. Four-Square (Cube) Decryption (Ciphertext Second Digraph: ┼E):
   a. Upper-Right Cube (Ciphertext First Letter: ┼) - Face 2, Row 4, Column 3
   b. Lower-Left Cube (Ciphertext Second Letter: E) - Face 1, Row 6, Column 3
   c. Upper-Left Cube (Plaintext First Letter: 4) - Face 1, Row 4, Column 3
   d. Lower-Right Cube (Plaintext Second Letter: d) - Face 2, Row 6, Column 3
   e. Plaintext – W_4d
   f. Upper-Right Cube Front Face (F1) - Face 2
   g. Upper-Right Cube Row Rotation: Row 4
   h. Lower-Left Cube Front Face (F1) - Face 1
   i. Lower-Left Cube Row Rotation: Row 6

8. Random Zigzag Transformation:
   a. Rotate the angle of $F1_{UR}$: $aUR_1 = (FIB_2 \bmod 4) * 90) = (21 \bmod 4) * 90 = 90$ degrees
   b. Rotate the R_{UR} to the right: $RR_{UR} = (FIB_2 \bmod 3) + 1 = (21 \bmod 3) + 1 = 1$ time
   c. Rotate the angle of $F1_{LL}$: $aLL_1 = (TRI_2 \bmod 4) * 90) = (4 \bmod 4) * 90 = 0$ degree
   d. Rotate the R_{LL} to the left: $RR_{LL} = (TRI_2 \bmod 3) + 1 = (4 \bmod 3) + 1 = 2$ times
   e. $FIB_n = FIB_{n+1} = FIB_{2+1} = FIB_3 = 34$
   f. $TRI_m = TRI_{m+1} = TRI_{2+1} = TRI_3 = 7$

9. Plaintext: W_4d

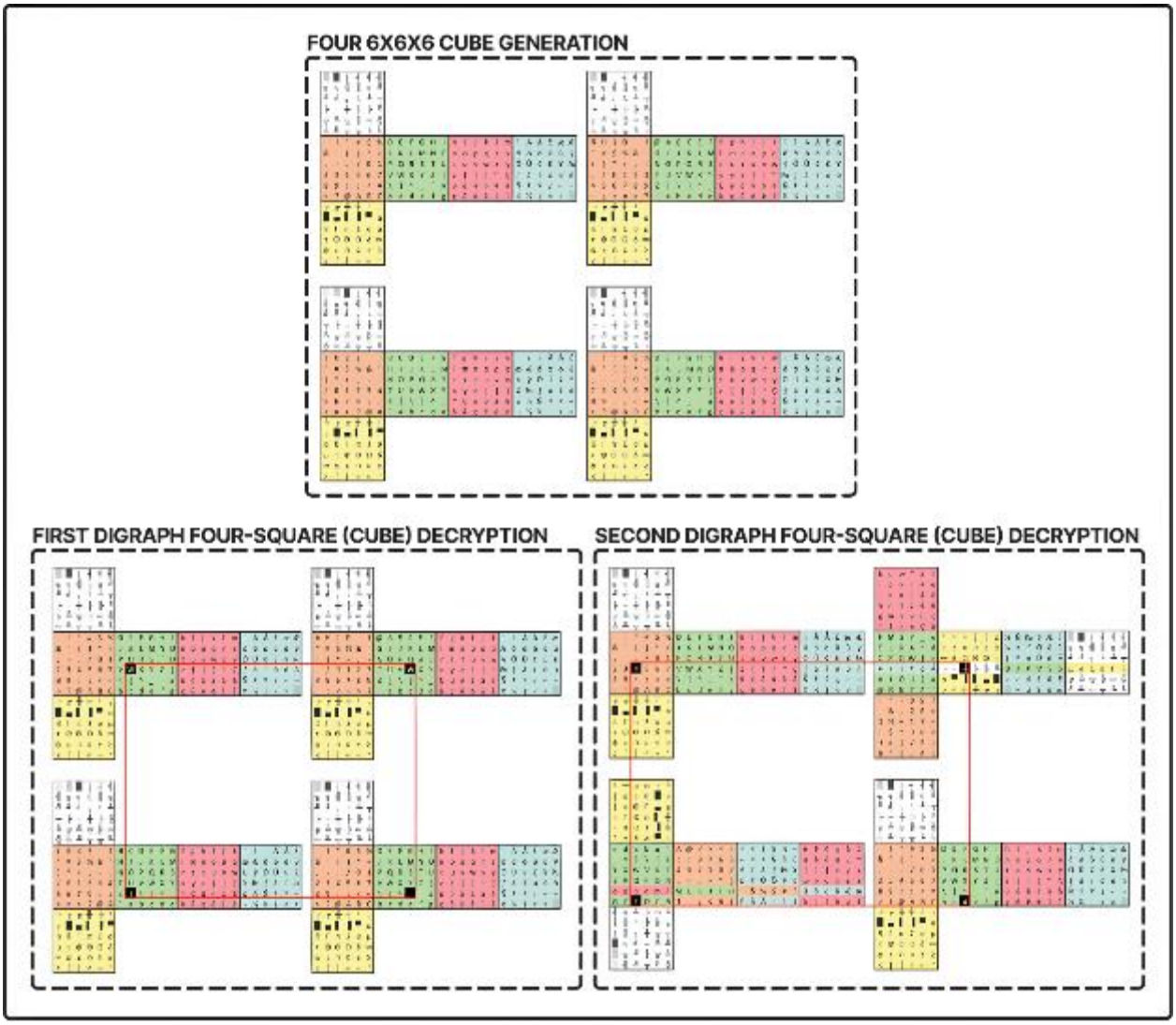

Figure 5 Decryption Process of Enhanced Four-Square with Zigzag Transformation Encryption Algorithm based on 3D Rubik's Cube Principle

2.10. Performance Evaluation and Testing

2.10.1. Frequency (Monobit) Test

The randomness of the binary sequence of the two random encrypted keys and ciphertext are evaluated using the Frequency (Monobit) Test by the NIST Special Publication 800-22 revision 1a [10]. According to the NIST Statistical Test Suite [10], the Frequency (Monobit) Test evaluates the numbers of zeros and ones in a binary sequence. The number of ones and zeroes should be almost the same for a binary sequence to be random. The randomness of a binary sequence is determined by its P-value. The P-value is computed and compared to the level of significance. The level of significance is 0.01. Based on the NIST Statistical Test Suite [10], the binary sequence is random if the P-value is greater than or equal to 0.01. Otherwise, the sequence is not random.

2.10.2. Runs Test

The randomness of the binary sequence of the two random encrypted keys and ciphertext are evaluated using the Runs Test by the NIST Special Publication 800-22 revision 1a [10]. Based on the NIST Statistical Test Suite [10], the Runs Test evaluates whether the number of runs of zeros and ones with different lengths is as expected for a random binary sequence. The randomness of a binary sequence is determined by its P-value. The P-value is computed and compared to the level of significance. The level of significance is 0.01. According to the NIST Statistical Test Suite [10], the binary sequence is random if the P-value is greater than or equal to 0.01. Otherwise, the sequence is not random.

2.10.3. Avalanche Effect

The avalanche effect is one of the evaluation methods used to test the strength of an encryption algorithm. The avalanche effect measures the change in ciphertext when the plaintext or the key is slightly changed. Even a single change of bit in a plaintext should significantly change the ciphertext. A cryptographic algorithm is a secure cryptographic algorithm if it has a strong avalanche effect; the avalanche effect should be more than 50% [11, 12]. According to Abikoye et al. [13], the avalanche effect could be computed by dividing the number of bits different in two ciphertexts and the total number of bits of the ciphertext and then multiplying it by 100, as shown in Equation (5).

$$Avalanche\ Effect\ (\%) = \frac{Number\ of\ bits\ differs\ in\ two\ ciphertexts}{Total\ number\ of\ bits\ in\ ciphertext}\ x\ 100 \quad (5)$$

The plaintext, two encryption keys, and two random numbers are the inputs of the encryption and decryption process of the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle. The avalanche effect of the ciphertext is evaluated to test the strength of the enhanced algorithm.

3. Results and discussion

This section will present and discuss the results of several evaluation methods and tests on the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle. The enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle is compared to the existing Four-Square with Zigzag transformation encryption algorithm by Elazzaby, El Akkad, and Kabbaj [6] to evaluate the robustness of the enhanced algorithm.

3.1. Matrix Comparison: Square Matrices VS. Cube Matrices

The inherent constraints of the existing Four-Square with Zigzag transformation encryption algorithm [6] are notably restrictive, as they demonstrate proficiency solely in encrypting and decrypting lowercase and uppercase letters from the English alphabet, as shown in Table 2. Its functionality stumbles when confronted with the inclusion of numbers and special characters, limiting its scope to a meager 50 characters that align with the conventional English alphabet. This constraint considerably impedes the algorithm's versatility and adaptability in handling a broader range of data types.

Table 2 Comparison of Allowed Characters in the Existing Four-Square with Zigzag Transformation Encryption Algorithm and Enhanced Four-Square with Zigzag Transformation Encryption Algorithm based on 3D Rubik's Cube Principle

	Existing four-square with zigzag transformation encryption algorithm [6]		**Enhanced four-square with zigzag transformation encryption algorithm based on 3d rubik's cube principle**	
	QUALIFICATION (/ - Accepted, X - Rejected)	NUMBER OF CHARACTERS	QUALIFICATION (/ - Accepted, X - Rejected)	NUMBER OF CHARACTERS
LOWERCASE LETTERS	/ (lowercase letters are transformed to uppercase letters)	25	/	26
UPPERCASE LETTERS	/	25	/	26
NUMBERS	X	0	/	10
SPECIAL CHARACTERS	X	0	/	154
TOTAL NUMBER OF CHARACTERS	**50**		**216**	

In contrast, the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle enhancement represents a significant stride forward in overcoming these limitations. This improved version can process letters, numbers, and special characters seamlessly throughout its cryptographic processes, as shown in Table 2. The capacity to encompass this expanded set of characters enhances the algorithm's utility in diverse applications where data may extend beyond the conventional alphabet.

The enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle substantially broadens the algorithm's functional repertoire by accommodating numbers and special characters as viable inputs for encryption and decryption. This expansion results in a notable increase in the available characters from 50 to 216, paving the way for more versatile and comprehensive cryptographic operations.

3.2. Comparison of Encryption and Decryption of Existing Four-Square Encryption Algorithm and Enhanced Four-Square Encryption Algorithm

In evaluating the seven distinct test cases employing the existing Four-Square with Zigzag transformation encryption algorithm [6], the encryption and decryption outputs reveal noteworthy patterns, as shown in Table 3. A consistent trend emerges despite the inherent variations in the plaintext across these diverse cases. In its encryption process, the existing Four-Square with Zigzag transformation encryption algorithm [6] predominantly transforms the plaintext into a recurring ciphertext string, identified as 'MUMBMFIVMS.'

This recurrent ciphertext, observed across various test cases, introduces a notable challenge in the decryption. The uniformity in the ciphertext poses a formidable obstacle when attempting to reverse-engineer the original plaintext from the encrypted data. This characteristic of the existing Four-Square encryption algorithm raises concerns about its ability to provide a sufficiently varied and secure encryption output.

In multiple instances, the recurrence of 'MUMBMFIVMS' underscores a potential vulnerability in the existing Four-Square with Zigzag transformation encryption algorithm's encryption methodology. The consequential difficulty in decrypting the original plaintext from this consistent ciphertext pattern highlights an area where the algorithm might benefit from enhancements.

Table 3 Encryption and Decryption of Existing Four-Square with Zigzag transformation Encryption Algorithm

Encryption		**Remarks**	**Decryption**	
Plaintext	**Ciphertext**		**Ciphertext**	**Plaintext**
Helloworld	Mumbmfivms	Equal	Mumbmfivms	Helloworld
Helloworld	Mumbmfivms	Not equal	Mumbmfivms	Helloworld
1234567890	Encryption error	N/a	N/a	N/a
!@#$% ^&*(	Encryption error	N/a	N/a	N/a
Helloworld	Mumbmfivms	Not equal	Mumbmfivms	Helloworld
Hello1world2	Mumbmfivms	Not equal	Mumbmfivms	Helloworld
Hello1!world2@	Mumbmfivms	Not equal	Mumbmfivms	Helloworld

Upon implementing the enhancement to the existing Four-Square with Zigzag transformation encryption algorithm [6] across the identical set of 7 test cases, a marked departure in results becomes evident compared to the outcomes produced by the unmodified algorithm. Introducing the enhancement brings about a notable diversity in the ciphertext generated for each distinct plaintext.

In stark contrast to the consistent 'MUMBMFIVMS' ciphertext pattern observed with the existing Four-Square with Zigzag transformation encryption algorithm [6], the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle demonstrates remarkable adaptability. Each unique plaintext undergoes encryption, yielding a correspondingly distinct ciphertext, as shown in Table 4. This variability in the encrypted output speaks to the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle capacity to tailor its encryption process to different inputs. It underscores an improved resistance to patterns that potential adversaries might exploit.

Furthermore, the decryption process following the use of the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle exhibits impressive efficacy. The ciphertext, characterized by its uniqueness for each test case, successfully reverts to the original plaintext.

Table 4 Encryption and Decryption of Enhanced Four-Square with Zigzag transformation Encryption Algorithm based on 3D Rubik's Cube Principle

ENCRYPTION		REMARKS AND RANDOM NUMBERS	DECRYPTION	
PLAINTEXT	**CIPHERTEXT**		**CIPHERTEXT**	**PLAINTEXT**
HELLOWORLD	8;-∩íα╡ µ±"	EQUAL (147, 6)	8;-∩íα╡ µ±"	HELLOWORLD
helloworld	<Ç=WFmj.ä`	EQUAL (162, 185)	<Ç=WFmj.ä`	helloworld
1234567890	(ÉL╔⊦ÿ0(k	EQUAL (83, 152)	(ÉL╔⊦ÿ0(k	1234567890
!@#$% ^&*(	N3~47ÇkΣ╕	EQUAL (34, 211)	N3~47ÇkΣ╕	!@#$% ^&*(
HelloWorld	=\[è■UWS£à	EQUAL (50, 98)	=\[è■UWS£à	HelloWorld
Hello1World2	;[VñçNK{ÿ2å╟	EQUAL (161, 141)	;[VñçNK{ÿ2å╟	Hello1World2
Hello1!World2@	>^[î^G9Jzú┴E─Ç	EQUAL (207, 17)	>^[î^G9Jzú┴E─Ç	Hello1!World2@

3.3. Frequency (Monobit) Test of Random Encrypted Keys and Ciphertext

Table 5 shows the result of the Frequency (Monobit) Tests of the two random encrypted keys and ciphertext of the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle. The table below shows the different character lengths of keys and ciphertext, the computed p-value, the remarks if the generated random encrypted keys and ciphertext are random or not, and the average p-value of all the random encrypted keys and ciphertext used in the test.

Table 5 Frequency (Monobit) Test of First Random Encrypted Key, Second Random Encrypted Key and Ciphertext

Character Length	First random encrypted key		Second random encrypted key		Ciphertext	
	P-value	**Remarks**	**P-value**	**Remarks**	**P-value**	**Remarks**
8	0.60	RANDOM	0.54	RANDOM	0.66	RANDOM
10	0.40	RANDOM	0.50	RANDOM	0.65	RANDOM
12	0.42	RANDOM	0.44	RANDOM	0.43	RANDOM
14	0.63	RANDOM	0.69	RANDOM	0.52	RANDOM
16	0.49	RANDOM	0.46	RANDOM	0.54	RANDOM
18	0.45	RANDOM	0.52	RANDOM	0.50	RANDOM
20	0.61	RANDOM	0.46	RANDOM	0.64	RANDOM
22	0.46	RANDOM	0.62	RANDOM	0.58	RANDOM
24	0.57	RANDOM	0.45	RANDOM	0.56	RANDOM
26	0.40	RANDOM	0.45	RANDOM	0.59	RANDOM
28	0.49	RANDOM	0.55	RANDOM	0.55	RANDOM
30	0.43	RANDOM	0.46	RANDOM	0.59	RANDOM
32	0.46	RANDOM	0.49	RANDOM	0.60	RANDOM
AVERAGE	**0.49**	**RANDOM**	**0.51**	**RANDOM**	**0.57**	**RANDOM**

The Frequency (Monobit) Test was performed ten times in each random encrypted key and each ciphertext. The p-value shown in the table below is the average p-value of the ten iterations in each random encrypted key and each ciphertext. The same plaintext and second key are used in each first key, the same plaintext and first key are used in each second key, and the same keys are used in each plaintext.

The range of the p-value of the first random encrypted key is between 0.40 to 0.63 and the average p-value is 0.49. The range of the p-value of the second random encrypted key is between 0.44 and 0.69 and the average p-value is 0.51. The range of the p-value of the ciphertext is between 0.43 and 0.66 and the average p-value is 0.57.

Based on the NIST-Statistical Test Suite [10], the level of significance is 0.01. Since the range of p-value and the average p-value of the two random encrypted keys and ciphertext is greater than 0.01, the binary sequence of the two random encrypted keys and ciphertext passed the frequency (monobit) test. Therefore, based on the Frequency (Monobit) test, the two random encrypted keys and ciphertext generated by the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle are random.

3.4. Runs Test of Random Encrypted Keys and Ciphertext

Table 6 shows the result of the Runs Tests of the two random encrypted keys and ciphertext of the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle. The table below shows the different character lengths of keys and ciphertext, the computed p-value, the remarks if the generated random encrypted keys and ciphertext are random or not, and the average p-value of all the random encrypted keys and ciphertext used in the test.

Table 6 Runs Test of First Random Encrypted Key, Second Random Encrypted Key and Ciphertext

Character Length	**First random encrypted key**		**Second random encrypted key**		**Ciphertext**	
	P-value	**Remarks**	**P-value**	**Remarks**	**P-value**	**Remarks**
8	0.43	RANDOM	0.51	RANDOM	0.48	RANDOM
10	0.44	RANDOM	0.66	RANDOM	0.60	RANDOM
12	0.68	RANDOM	0.44	RANDOM	0.50	RANDOM
14	0.38	RANDOM	0.68	RANDOM	0.52	RANDOM
16	0.52	RANDOM	0.66	RANDOM	0.54	RANDOM
18	0.56	RANDOM	0.53	RANDOM	0.49	RANDOM
20	0.51	RANDOM	0.49	RANDOM	0.32	RANDOM
22	0.43	RANDOM	0.46	RANDOM	0.25	RANDOM
24	0.66	RANDOM	0.65	RANDOM	0.41	RANDOM
26	0.49	RANDOM	0.43	RANDOM	0.35	RANDOM
28	0.33	RANDOM	0.53	RANDOM	0.45	RANDOM
30	0.38	RANDOM	0.42	RANDOM	0.31	RANDOM
32	0.32	RANDOM	0.40	RANDOM	0.27	RANDOM
AVERAGE	**0.47**	**RANDOM**	**0.53**	**RANDOM**	**0.42**	**RANDOM**

The Runs Test was performed ten times in each random encrypted key and each ciphertext. The p-value shown in the table below is the average p-value of the ten iterations in each random encrypted key and each ciphertext. The same plaintext and second key are used in each first key, the same plaintext and first key are used in each second key, and the same keys are used in each plaintext.

The range of the p-value of the first random encrypted key is between 0.32 to 0.68 and the average p-value is 0.47. The range of the p-value of the second random encrypted key is between 0.40 to 0.68 and the average p-value is 0.53. The range of the p-value of the ciphertext is between 0.27 and 0.60 and the average p-value is 0.42.

Based on the NIST-Statistical Test Suite [10], the level of significance is 0.01. Since the range of p-value and the average p-value of the two random encrypted keys and ciphertext is greater than 0.01, the binary sequence of the two random encrypted keys and ciphertext passed the Runs Test. Therefore, based on the Runs Test, the two random encrypted keys and ciphertext generated by the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle are random.

3.5. Repeating the Same Characters Multiple Times in the Plaintext

Table 7 shows the comparison between the encrypted plaintext (ciphertext) of the existing Four-Square with Zigzag transformation encryption algorithm by Elazzaby, El Akkad, and Kabbaj [6] and the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle when same characters are repeated multiple times in the plaintext. The table below shows the different plaintext the algorithm encrypted, the number of times the word is repeated in the plaintext, and the equivalent ciphertext of each plaintext encrypted using the two algorithms. Five different plaintexts are used to compare the ciphertext generated by the two algorithms. The number of times the word is repeated in the plaintext ranges from 26 to 30 times. The same keys are used in each plaintext.

The generated ciphertext of the plaintext is encrypted using the existing Four-Square with Zigzag transformation encryption algorithm by Elazzaby, El Akkad, and Kabbaj [6], which contains unique characters when repeated 25 times in plaintext. However, a pattern can be observed when the characters are repeated more than 25 times in plaintext. The 26th digraph and the succeeding digraph of the ciphertext are the same as the first digraph, second digraph, third digraph, and the following digraphs. Therefore, the ciphertext generated by the existing algorithm can be decrypted easily by repeating the characters more than 25 times in the plaintext and observing the pattern of the generated ciphertext when the two keys are known.

On the other hand, the generated ciphertext of each plaintext encrypted using the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle contains random characters despite the characters being repeated in the plaintext more than 25 times. Therefore, the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle generates a random ciphertext and cannot be decrypted by repeating the characters in the plaintext multiple times and observing the pattern of the generated ciphertext, even when the two keys are known.

Table 7 Comparison of Existing Four-Square with Zigzag Transformation Encryption Algorithm and Enhanced Four-Square with Zigzag Transformation Encryption Algorithm based on 3D Rubik's Cube Principle when Same Words are Repeated Multiple Times in the Plaintext

Plaintext	**Number of times the words is repeated**	**Ciphertext**	
		Existing four-square with zigzag transformation encryption algorithm [6]	**Enhanced four-square with zigzag transformation encryption algorithm based on 3d rubik's cube principle**
OnOnOnOnOnOnOnOnOn OnOnOnOnOnOnOnOnOn OnOnOnOnOnOnOn	25	GHEKDLCNAOSPKQISBURVUW OYFMZAYTXRWIVXTZQGPBNC MDLEHF	hEfS≡- Ç7╟╞] ╒≤Cê8y≡a6Θf╕ ▒:Ä≤╟╞áf≤ªa≡y ╒óh ê≡Q≡SÇáf▄é]Äà
AMAMAMAMAMAMAMA MAMAMAMAMAMAMAM AMAMAMAMAMAMAMA MAMAMAM	26	ODFEZFYHXKWLVNTOQPPQNS MULVHWGYEMDACTARSIKXIZ BGRBUCOD	_£±æßDßτ±ú⌐æëDO ▌tæ╜ DOæëp⌐.«£'æ' ║Ö£äDëæO*tW'D╥æë ▌⌐£«æ
Dogsdogsdogsdogsdogsd ogsdogsdogsdogsdogsdo gsdogsdogsdogsdogsdogs dogsdogsdogsdogsdogsd ogsdogsdogsdogsdogsdo gs	27	RHKOOLBQZOUUXQFWVUYMQ WWTNMTILTPZGIMBDZHDAB EFKDCKBFSNUKIPFNRSYPOVW SZYTVXAPYVRMAQXHRNGEXL CCGGESCDHIEALRHKOOLBQ	âB┤╣¬K÷Ω┘ s╙ö¬Véíâ╓<^üΓ«^¬Vk▌ \φ; ╣┘ N╥╨¬h5╨â ┌çS╡│»^âs⌐Å¬h`0ü£ê8¬┴ ╪╨âÖax╡ h╙Å¬%ì^\B╥öü╓òx¬15xâKç8 Æh;╣âV±0¬┴»╙╡ φ÷^
CubeCubeCubeCubeCube CubeCubeCubeCubeCube CubeCubeCubeCubeCube	28	BORTUQOIFUZZYWXBWMVDT TQFPINKMZLNHBGPEDDSCFAV SKKYINBARPUROSFXZVYGXYW	â%è]B£^╪â ╒'«Ä=N8âS¢çÄ7'╥î&^╙ZΣè[f %╛ ÷╟╞σ≤╪fOÜ[╒eué\àd:B¼9c\ ╒ué▒ └N

CubeCubeCubeCubeCube CubeCubeCubeCubeCube CubeCubeCube		CVATEQRPHNXMLLGHOGCEQD ECUAHSWKLIMBORTUQOIFUZ Z	ç\╨»╥▒tu8fu9═±Nd÷@=Γ╪╟¼¼¢[@▒Ü ╥ ╒t'ê7O>±Bσ¬÷7%'êZ7N4
SECONDSECONDSECOND SECONDSECONDSECOND SECONDSECONDSECOND SECONDSECONDSECOND SECONDSECONDSECOND SECONDSECONDSECOND SECONDSECONDSECOND SECONDSECONDSECOND SECONDSECOND	29	TTUHEINXZNAGHBWQIDDEQV UHSKMMZNBOGRWQOSCZQVY WKCMMVARFGRPIFLCZLGXPK CEDTURFAHNYFLINHTXPUQDX TUZVSBNYWMBEHTQROKDXM ZYOSBGCVSBECFPWOKKLLAYO RPEIVSFUAGPWXYIDLATTUHE INXZNAGHBWQIDDEQVUH	H?╧ÆZ═,╟ φ»c5Q¬≥Æ?⌐m6 ╔86⌐H▒:êZ ð, ▌^<c═Q¥@ê?⌐m▒; ╞6[H≥Y»Zf,^åΩcB Qeñ»?5m ▌¬à6hH ╔ï¼Z┤,Θ ╔mcÄQ≥≥¼? ⌐m ▌└╞6îH ╒:óZº,┼▒zc╦Q▒╧ó?hm- ÇÆ65Hn╬HZ3,e¬ ╚c]Q╬ñH?⌐mnn<6╪H- ïUZ▓,I^Ãc∩Q╟YU?▓mT; ⌠ 6]H?_╨Z╪
lengthlengthlengthlength lengthlengthlengthlength lengthlengthlengthlength lengthlengthlengthlength lengthlengthlengthlength lengthlengthlengthlength lengthlength	30	LMDBMDERSEGHAZBKCNICOO KQUFYSRVZLVWFMWPPAXRQ ULITZMYEGNCGTADHFCXIHDL KBUNSPREZQBUFKWVOYXOQ MYTTSMRVXNWGZPBHACCLE DIKFEKSGRLAOBDFPISOHXUU WYNTYZAVQNTWIPVHXQGLM DBMDERSEGHAZBKCNICOOKQ UFYSRV	Aän╧:56■1[:Ñf."à:¬6╖ |a^╨fäna^°Al1fà° î╖ "½à≤6■|aà5fänåà[AÃ1εàÇîl"aàåA╖ |+ ¬°îçn≤¬▒f─1ï:▒6╖ "½:+AÃ|¬: ╒îçn╨:Afy 1[:f6─"à:¬fÃ|<^▒6än¬^[î─1Aà╨AÃ"åàaf y|╨à[6än:àεîy1[à5A."½à`î╖ |a¬▒Abn+¬f 6.1ε:°

3.6. Avalanche Effect of the Enhanced Four-Square with Zigzag transformation Encryption Algorithm based on 3D Rubik's Cube Principle

Table 8 shows the comparison between the avalanche effect of the existing Four-Square with Zigzag transformation encryption algorithm by Elazzaby, El Akkad, and Kabbaj [6] and the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle. The table shows the different sets of plaintext, its equivalent ciphertext, random numbers used to randomize the key and rotation of the cubes, the avalanche effect of each set of plaintext, and the average avalanche effect of the two algorithms.

As the existing Four-Square with Zigzag transformation encryption algorithm {6] accepts lowercase and uppercase letters only, three sets of almost the same plaintext where only one character is changed are used to compute and compare the avalanche effect of the two algorithms. In the first set of plaintext, one of the lowercase letters of the other plaintext is changed to an uppercase letter. In the second set of plaintext, one of the lowercase letters of the other plaintext is changed to a different lowercase letter. In the last set of plaintext, the uppercase letter of the other plaintext is changed to a different uppercase letter. The same sets of almost the same plaintexts are encrypted in both algorithms, and the same keys are used in each plaintext.

The avalanche effect of the existing Four-Square with Zigzag transformation encryption algorithm by Elazzaby, El Akkad, and Kabbaj [6] in all different sets of plaintext tested is consistently 4.69%. Consequently, the average avalanche effect of the existing Four-Square with Zigzag transformation encryption algorithm [6] is also 4.69%. The existing Four-Square with Zigzag transformation encryption algorithm [6] does not satisfy the condition that the avalanche effect of a secure encryption algorithm should be more than 50%. This denotes that the existing Four-Square with Zigzag transformation encryption algorithm [6] is considered not a secure encryption algorithm.

On the other hand, compared to the existing Four-Square with Zigzag transformation encryption algorithm [6], the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle had a promising result, the avalanche effect in different sets of plaintext tested range between 51.39% to 54.17% and the average avalanche effect is 52.78%. The enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle satisfies the condition that the avalanche effect of a secure encryption algorithm should be more than 50%. Thus, the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle is a secure encryption algorithm.

Table 8 Comparison of Avalanche Effect of the Existing Four-Square with Zigzag Transformation Encryption Algorithm and Enhanced Four-Square with Zigzag Transformation Encryption Algorithm based on 3D Rubik's Cube Principle

Plaintext	**Existing four-square with zigzag transformation encryption algorithm [6]**		**Enhanced four-square with zigzag transformation encryption algorithm based on 3d rubik's cube principle**		
	Ciphertext	**Avalanche effect (%)**	**Ciphertext**	**Random number**	**Avalanche effect (%)**
Graphics	ELRDEXZQ	4.69%	gG"$åe4 ▌	52, 151	52.78%
GraFhics	ELZMEXZQ		dHΩ≤k≤┘ #	53, 65	
Keyboard	GKVKDSNK	4.69%	C] ╞>çörI	178, 147	54.17%
Ceyboard	CBVKDSNK		^;-ë≈k╖ #	129, 203	
Software	SEALSIPK	4.69%	oE5Φ⌡ LH▓	186, 74	51.39%
Softvare	SEALSBPK		nJaQ¢Ç╖L	46, 111	
AVERAGE	**4.69%**		**52.78%**		

Table 9 shows the avalanche effect of the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle. The table shows the different sets of plaintext, its equivalent ciphertext, random numbers used to randomize the key and rotation of the cubes, the lowest avalanche effect of each set of plaintext, the highest avalanche effect of each set of plaintext, and the average avalanche effect of the algorithm.

Table 9 Avalanche Effect of the Enhanced Four-Square with Zigzag Transformation Encryption Algorithm based on 3D Rubik's Cube Principle

PLAINTEXT	**LOWEST**		**HIGHEST**	
	CIPHERTEXT	**AVALANCHE EFFECT (%)**	**CIPHERTEXT**	**AVALANCHE EFFECT (%)**
Graphics	dF+ë∞╟«	33.33%	hH!αç≤ù1	52.78%
GraFhics	cHúCS═¼j		gGδ6╨;Iε	
Keyboard	F\n ╒╒Vjï	36.11%	F\∞╥╖úE╧	51.39%
Ceyboard	^<Ω^;nX"		]<+»≈~╖"	
Software	qF<τP∩m»	33.33%	nG4+÷ÆnI	54.17%
Softvare	pD:σO£╚æ		qF╦Φ¢∩▌ ▒	
Zero One	S_ói6▓ú╬	33.33%	RaÑÅ ┌Ñ!B	55.56%
$ero One	?<úâ&⌐ñà		<:ôhâ╪/╡	
Device1234	>\PtsÖ÷╧ ╒^ùù	36.36%	>[ú¥¢höTτ'	53.41%
Device8234	?\Ñ£╨ ⌠ GèQP		?\Ædd ⌠ m)R╦	
Hexa_decimal	?bτéáç5╩a▒Åc	41.35%	?bsªΘfZmLbC▒	51.92%
Hexa:decimal	A`τöTû3£ç╔m╣		@^τ~Sy&½ç8ôe	
Computer	ä!&Æδ3èj	31.94%	â"v1[z½Ω	52.78%
Co3puter	ä!,|∞y;Æ		ä!+B-æUV	
Question	oJ;∩|╤ÇV	31.94%	qL·╦q≤43	54.17%
Que%tion	oLaA0 ⊢╦2		oIü⌡ ╨ ╒═══	
Javascript	GWσ╣╡ =╕ `⌐T	37.50%	GW(G▌ è≡i¬ÿ	53.41%

7avascript	R6s φb■C¿p		T5τ╛ VSYô≥w	
Filesystem86	b= æ▒ê¬Σ┤ Lka	35.58%	d@‖ çφe┘ P;_±P	51.92%
Filesystem&6	b?:ôbÜh(½äñ		`?bè$£ª_¡î2O	
AVERAGE	**35.08%**		**53.15%**	

Ten sets of almost the same plaintext where only one character is changed are used to compute the avalanche effect of the algorithm. The only character changed in the other plaintext can be another lowercase letter, an uppercase letter, a number, or a special character. The same keys are used in each plaintext. Each set of plaintext is encrypted five times. The avalanche effect is computed in each iteration in each set of plaintext. The table below shows the lowest and highest avalanche effect in each set of plaintext.

The avalanche effect of the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle has its fluctuating outcome due to its random number generator (RNG) that is incorporated into the key generation. The lowest and highest outcomes ranged between 33.33% to 54.17%, resulting in an average of 35.08% for the lowest avalanche effect and 53.15% for the highest avalanche effect. The enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle reaches the condition that the avalanche effect of a secure algorithm should be more than 50%. Therefore, the enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle is a robust encryption algorithm.

4. Conclusion

The study successfully enhanced Four-Square with Zigzag transformation encryption algorithm based on 3D Rubik's cube principle, bolstering its security and applicability. By transforming the 5x5 matrix into a dynamic 6x6x6 cube, accommodating 216 characters, including uppercase and lowercase letters, numbers, and special characters, significant improvements were achieved. Statistical randomness tests, including the Frequency (Monobit) and Runs tests, confirmed the robustness of the encryption method, surpassing established thresholds. The evaluation of the avalanche effect demonstrated a substantial enhancement compared to the existing algorithm, meeting the minimum requirement of a 50% avalanche effect. This research addresses the limitations of the Four-Square cipher, contributing to cryptography with a versatile and secure algorithm. The findings hold promise for enhancing digital communication security, and ensuring confidentiality, integrity, and authenticity of sensitive information. This study lays the groundwork for future research in encryption techniques, fostering further advancements in the field.

Compliance with ethical standards

Acknowledgments

We express our sincere gratitude to Elazzaby F., El Akkad N., and Kabbaj S. for providing inspiration for our research. Their pioneering contributions to Cryptography have laid a robust foundation for our study. The depth of insight we gained from their work has enriched our theoretical understanding. The methodologies and innovative approaches employed by them have served as a beacon, influencing the decisions we made during our study.

Furthermore, we would also like to extend our gratitude to our adviser and mentors for their continuous support and motivation throughout the process, which played a pivotal role in guiding the practical aspects of our research. Also, our appreciation goes out to our colleagues and friends for standing by us with their support and encouragement. The quality of our study owes much to their constructive feedback, stimulating discussions, and moral support.

Lastly, we would also like to express gratitude to the Department of Science and Technology (DOST) for providing financial assistance, which has been instrumental in the completion of this study. Additionally, we extend a special thank you to our parents for their continuous financial backing, guidance, encouragement, and lifelong inspiration. Their unwavering presence and guidance were invaluable throughout our journey.

Disclosure of conflict of interest

The authors declared that there is no conflict of interest to be disclosed.

References

[1] Aishwarya J, Palanisamy V, Kanagaram K. An extended version of four-square cipher using 10 x 10 matrixes. International Journal of Computer Applications. 2014;97(21):9-13. doi: 10.5120/17129-7615.

[2] Hossain Biswas MdS, Ali A, Rahman M, Khaled Sohel Md, Maruf Hasan Md, Sarkar K, et al. A systematic study on classical cryptographic cypher in order to design a smallest cipher. International Journal of Scientific and Research Publications (IJSRP). 2019;9(12). doi: 10.29322/ijsrp.9.12.2019.p9662.

[3] Arroyo JC, Reyes AR, Delima AJ. A novel ASCII code-based polybius square alphabet sequencer as enhanced cryptographic cipher for cyber security protection (APSAlpS-3CS). International Journal of Advanced Computer Science and Applications. 2020;11(7). doi: 10.14569/ijacsa.2020.0110715.

[4] Arroyo JC, Dumdumaya CE, Delima AJ. Polybius square in cryptography: a brief review of literature. International Journal of Advanced Trends in Computer Science and Engineering. 2020;9(3):3798–808. doi: 10.30534/ijatcse/2020/198932020.

[5] Akanksha D, Samreen R, Niharika GS, Shruthi A, Kiran MJ, Venkatramulu S. A hybrid cryptosystem based on modified vigenere cipher and polybius cipher. EPRA International Journal of Research and Development (IJRD). 2022;7(6):113-9.

[6] Elazzaby F, El Akkad N, Kabbaj S. A new encryption approach based on four-square and Zigzag Encryption (C4CZ). Embedded Systems and Artificial Intelligence. 2020;589–97. doi:10.1007/978-981-15-0947-6_56.

[7] ASCII Table for Code page 437: ASCII Code Reference [Internet]. [place unknown]: [publisher unknown]; © 2005-2024 [updated 2024; cited 2023 Dec 6]. Available from https://www.ascii-code.com/CP437.

[8] Alfke P. Efficient shift registers, LFSR counters, and long pseudo-random sequence counters. Xilinx Application Note XAPP 052. 1996:1-6.

[9] Manliclic GM, Lamac KA, Regala RC, Blanco MC. Improving the extended 10x10 polybius square key matrix for playfair bifid and polybius cipher. United International Journal for Research & Technology. 2023;4(07):290-5.

[10] Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert A, Dray J. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Gaithersburg, MD: National Institute of Standards and Technology, Special Publication (NIST SP); 2010.

[11] Echeverri C. Visualization of the Avalanche Effect in CT2 [Bachelor's thesis]. University of Mannheim;2017.

[12] Mohamed K, Mohammed Pauzi MN, Mohd Ali FH, Ariffin S. Analyse On Avalanche Effect In Cryptography Algorithm. European Proceedings of Multidisciplinary Sciences. 2022.

[13] Abikoye OC, Haruna AD, Abubakar A, Akande NO, Asani EO. Modified advanced encryption standard algorithm for information security. Symmetry. 2019 Dec 5;11(12):1484.