International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET2310222

Text Encryption Using Improving Key of Hi Sec Algorithm Using Rubik's Cube

Israa Shihab Ahmed
Informatics Institute for Postgraduate Studies, Iraqi Commission for Computers and Informatics, Baghdad Iraq

ARTICLEINFO

Article History:
Accepted: 05 March 2023
Published: 30 March 2023

Publication Issue
Volume 10, Issue 2
March-April-2023

Page Number
218-226

ABSTRACT

Several techniques have been developed to generate keys using Rubik's Cube, which has been researched as a potential source of randomness for cryptography. This study suggests an innovative technique for creating a key for the Hi Sec algorithm using a Rubik's Cube image. The process entails creating a binary key for text encryption by leveraging the cube's color information. The Rubik's Cube principle, which includes choosing a subset of cube faces to generate the key, serves as the foundation for the key creation process. Experimental findings indicate that the proposed approach can produce safe keys that are resilient to a variety of assaults. The technique offers a fresh and effective way to generate keys for the Hi Sec algorithm, which may be used in several secure communication systems. The proposed method was programmed in VisualBaic.Net 2012.

Keywords: Rubik's Cube, cryptography, key generation, Hi Sec algorithm, bitwise operations, binary representation.

I. INTRODUCTION

The practice of encrypting data into cipher text, a distinct format that can be used to secure it. The message cannot be deciphered (or converted into plaintext) without a key. The ciphers are broken via cryptanalysis without the usage of a key. Electronic security is becoming more crucial as Internet-based communication grows more common. Messages, credit card data, and corporate data are protected by cryptography against intruders and their attacks [1]. Public key cryptography and private key cryptography are the two basic categories of cryptographic algorithms. The secret key must be shared by both the sender and receiver to encrypt the data and decrypt the data using private key cryptography because it uses identical encryption and decryption keys [2]. One of the most used algorithms in private key encryption is the Advanced Encryption Standard (AES) algorithm, followed by Triple DES, Data Encryption Standard (DES), and Public Key Encryption, which encrypts and decrypts data using two different keys. Using a public key to encrypt data and a private key to decrypt it allows for secure communication even if the public key is intercepted by a third party (e.g RSA algorithm etc). Figure 1 shows many categories for classifying methods in cryptography [3].

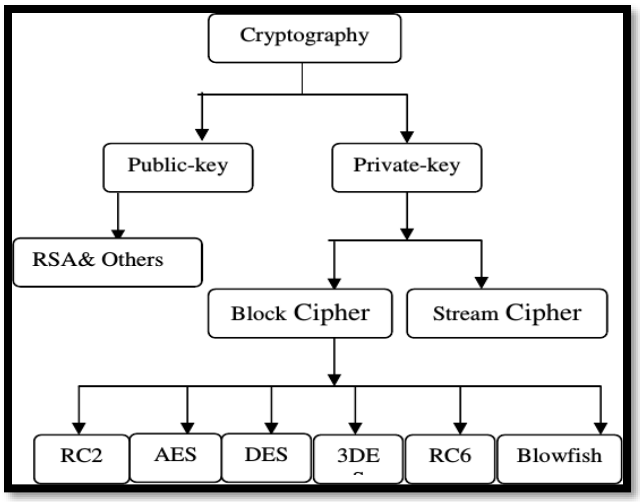


Figure 1- Classifying Methods in Cryptography.

In this study, a symmetric-key cryptography technique called the Hi Sea Algorithm uses a Rubik's Cube to generate keys. The technique was developed to counteract risks posed by computers and offers a high level of safety. The following sections make up the remainder of this paper, the suggested encryption technique based on the Rubik's cube idea is described in Part 2. The concepts of the suggested technique are outlined in Part 3, its implementation is presented in Section 4, its statistical tests are explained in Section 5, its result analyses are explained in partition 6, and conclusions are explored in partition 7.

II. Algorithms Overview

In this section, we will explain the most important algorithms that were used in this research, which are the Rubik's cube and the HI Sea Algorithm.

A. Rubik's Cube Technique

It is an intellectually and technically challenging 3D-building game. The idea was developed in 1975 by Erno Rubik. One of the most played games, it is estimated that over 350 million units have been sold worldwide as of 2009 [4]. Since color images are made up of millions of pixels, each of which can choose a random color from a palette of more than 16 million colors, the cube that is an order list with 54 fields (that is, 6x9 values) can be broken into a finite amount of the time. When basic operations (such as face rotation, cube rotation, and their combinations) are applied, it turns into an interesting way to swap pixels in an image [5, 6, 7, 8]. The Rubik's cube method is used to scramble the positions of the image pixels, which is a crucial step in adding a high degree of permutation. Numerical arrays are frequently found in digital photos. Grayscale images are represented as 2D matrices as opposed to colorful images, such as RGB images, which are represented as 3D matrices. Grayscale images can be employed with the classic Rubik's Cube technique even if color images make up the majority of the images we want to deal with [6] [7]. Figure 2 illustrates Rubik's Cube and RGB plane [9].

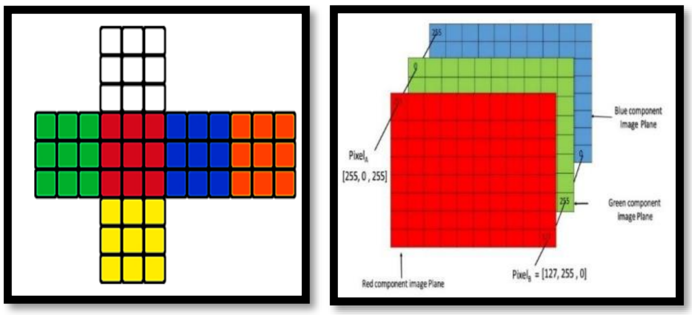

255
0
Pixel$_A$
[255, 0 , 255]
Blue component Image Plane
0
255
Green component image Plane
Red component image Plane
Pixel$_B$ = [127, 255 , 0]

Figure 2: Rubik Cube & RGB plane [9].

B. Hi Sec Algorithm:

Hi Sec employed an alternative way for bit permutation while maintaining the present properties. With certain tweaks, the Hi Sec algorithm's structure resembles that of feistily [10] [11Hi Sec uses an 80-bit key and 64-bit plain text. Operations such as replacebox, bit-swap, XOR, rotation, and key update are used in each of the 15 rounds. Moreover, in the last round, the key is XOR and the cipher text. The HISEC is divided into four layers, which are as follows [12]:

• First Layer: The 64-bit plaintext and 80-bit key are XOR in this layer. The plaintext is split into two sections. The outputs of each 32-bit component from XOR serve as the second layer's inputs (Substitution box).

• The second layer is the one that matters the most. This gives the algorithm nonlinearity and results in the confusion property. It divides its 16 4-bit S-boxes into two halves, with eight S-boxes in each. The third layer's input will come from this layer's output (bit permutation. An S-box is repeated 16 times on this layer as well. The qualities of a good S-box mirror those of the S-box. The S-box values are displayed in table (1).

Tables 1. Values in the S-box [12]

X	0	1	2 3	4	5	6	7	8	9	A	B	C	D	E	F
S(X)	F	C	2 7	9	0	5	A	1	B	E	8	6	D	3	4

• Third Layer: This layer produces dispersion, a key element of any successful encryption method. This bit permutation technique works with two sides, each of which has 32 bits.

• Fourth Layer: The equation's two sides are rotated in this layer. Before XOR the left and right 32-bits, rotate the left 32-bit. On the left, the result will remain 32-bit. After rotating the right 32-bit and XOR it with the new left 32-bit, the result must be stored in the right 32-bit.

• The key schedule is every encryption method's last and most important element, K1, K2, K3, K4, and K79 makes up -bit master key size. Here is how the key update or key schedule works: Rotate the master key left P bits, where P has a starting value of 13. The master key is now equal to the master key minus 13, and in the ensuing round, P will rise by 2. Rotate the master key left P bits, where P has a starting value of 13. Following that, P will rise by 2 for the subsequent round, and the master key will be equal to the master keyless1than 3. Only 64 bits are used in the encryption technique, while the master key is 80 bits. The master key's 64-bit maximum length is required by the encryption technique [12]. Figure 3 depicts the four layers of Hi Sec [13].

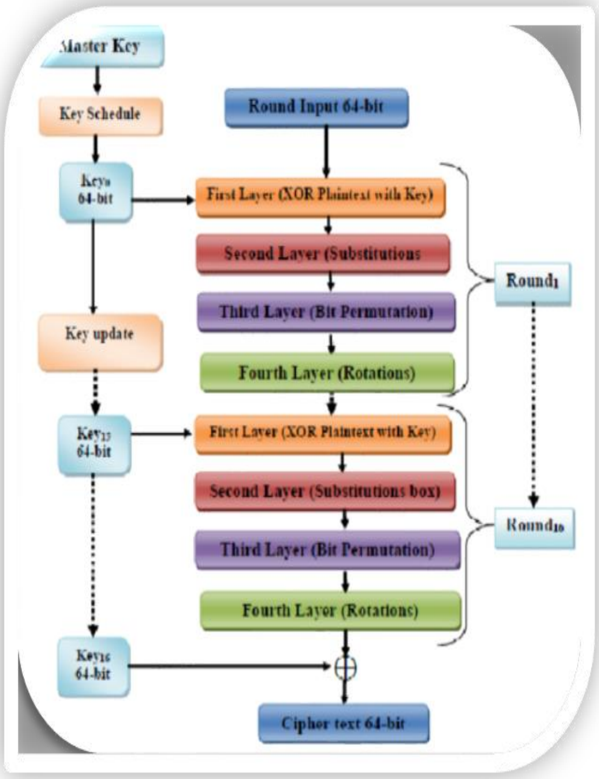

Master Key
Key Schedule
Round Input 64-bit
Keys 64-bit
First Layer (XOR Plaintext with Key)
Second Layer (Substitutions
Round1
Third Layer (Bit Permutation)
Key update
Fourth Layer (Rotations)
Key15 64-bit
First Layer (XOR Plaintext with Key)
Second Layer (Substitutions box)
Round15
Third Layer (Bit Permutation)
Fourth Layer (Rotations)
Key16 64-bit
Cipher text 64-bit

Figure (3). Hi Sec Algorithm Layers [13].

III. The Proposed Method's Principles:

The HI Sea Algorithm generates encryption keys using a novel method based on Rubik's Cube. The following steps are involved in the key generation process:

1. Choose a random Rubik's Cube configuration: The key generator randomly scrambles a Rubik's Cube to create a unique configuration. This configuration serves as the basis for the encryption key.

2. Apply a fixed set of movements: To further jumble the Rubik's Cube configuration, the key generator uses

a fixed set of moves. These actions aim to increase the key's complexity and unpredictability.

3. Identify the key: The key is taken out of the final Rubik's Cube configuration by the key generator. To do this, a preset mapping strategy is used to map the Rubik's Cube colors onto a binary string. The resulting binary string serves as the encryption key. As shown in figure 3 key generating from Rubik's Cube.

Here is a description of the first random image that was used to generate the keys.

Step 1: the input image's red, green, and blue planes are retrieved. Each plane is then split into 8 × 8-inch blocks for each color.

Step 2: The following operations are performed on each pixel in the Red plane's blocks:

i) The value of each pixel's binary representation is converted into digits. Then, depending on how many 1s and 0s there are in the binary representation, each pixel is changed into a 0 or a 1. The pixel value is transformed to 1 if the binary value contains more ones (1s) than or equal to 0; otherwise, it is turned to 0.

ii) If the block size is larger than or equal to the obtained value, a value is transformed to 1; otherwise, it is divided by 0.

iii) Then the row of each block is taken after this action is completed, and the resulting 8-bit binary value is then converted to a decimal number. It is possible to create an array using an array containing the decimal values generated for each block. Again, steps I through (3) are performed by the green and blue planes.

Step 3: Once the updated values for each red, green, and blue level are collected, the average of the three levels is provided as the initial vector for the key generation procedure using the Hi sec method. Figure 4 uses the corkscrew made using a Rubik's Cube to illustrate the procedure.

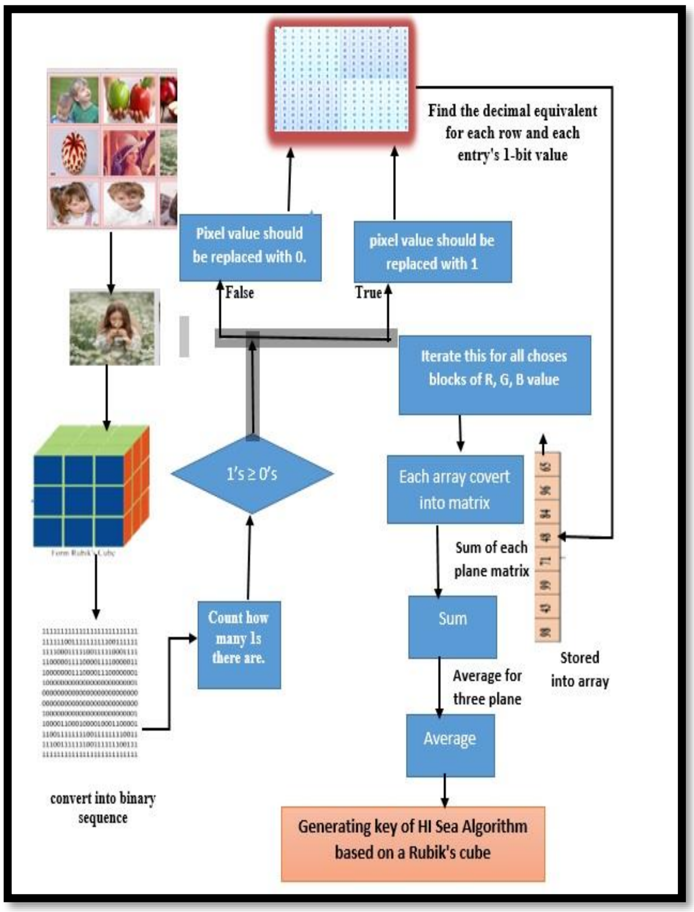

Find the decimal equivalent for each row and each entry's 1-bit value
Pixel value should be replaced with 0.
pixel value should be replaced with 1
False
True
Iterate this for all choses blocks of R, G, B value
1's ≥ 0's
Each array covert into matrix
Sum of each plane matrix
Count how many 1s there are.
Sum
Stored into array
Average for three plane
Average
convert into binary sequence
Generating key of HI Sea Algorithm based on a Rubik's cube

Figure 4. Key Generating from Rubik's Cube.

The key generation process ensures that each key is unique and unpredictable, providing a high level of security. Additionally, the use of a physical object such as a Rubik's Cube adds an extra layer of security by making the key generation process difficult to replicate or reverse engineer.

In general, the key generation process of the Hi Sea algorithm, which is based on the Rubik's Cube, is a unique and innovative method that provides encryption keys with a high level of security.

Finally, the improving Hi Sea Algorithm to encrypt a message isshown in Figure 5 shows the proposed method's general layout.

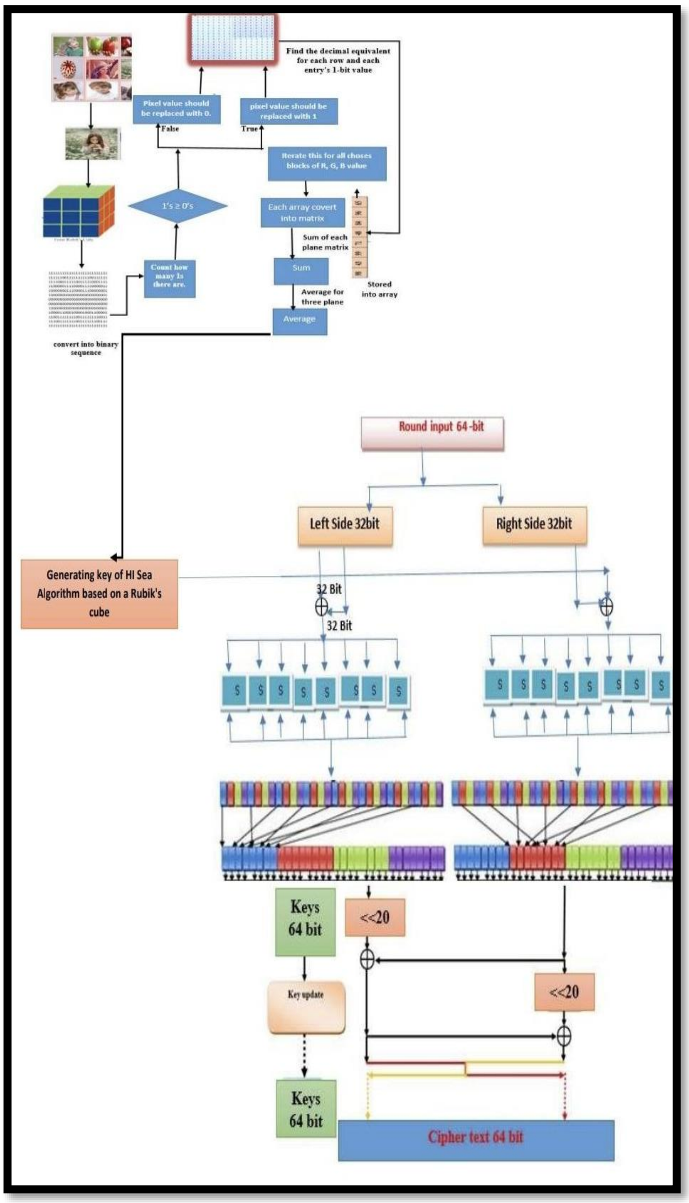

Find the decimal equivalent for each row and each entry's 1-bit value
Pixel value should be replaced with 0.
pixel value should be replaced with 1
False
True
Iterate this for all choses blocks of R, G, B value
1's ≥ 0's
Each array covert into matrix
Sum of each plane matrix
Count how many 1s there are.
Sum
Average for three plane
Stored into array
Average
convert into binary sequence
Round input 64-bit
Left Side 32bit
Right Side 32bit
Generating key of Hi Sea Algorithm based on a Rubik's cube
32 Bit
32 Bit
Keys 64 bit
<<20
Key update
<<20
Keys 64 bit
Cipher text 64 bit

Figure 5. The Suggested Approach's General Layout.

IV. Implementation of The Suggested Approach:

This partition will explain how to implement the suggested way in the program using the procedures shown in Figure 4, which demonstrated how to create the Hi sec algorithm key using a Rubik's cube, then encode and decode the original message using the improved algorithm method. Figure 6 depicts the suggested method's interface.

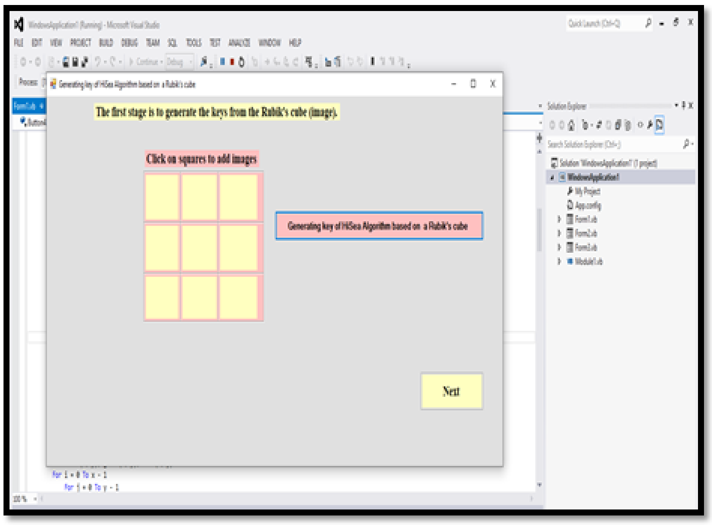

Generating key of HiSea Algorithm based on a Rubik's cube
The first stage is to generate the keys from the Rubik's cube (image).
Click on squares to add images
Generating key of HiSea Algorithm based on a Rubik's cube
Next

Figure 6. window main to work proposed.

As shown in figure 7, the Hi sec algorithm key is obtained in the first phase. One movement of the Rubik's Cube provided a picture from which I worked to generate the key after it had been generated randomly.

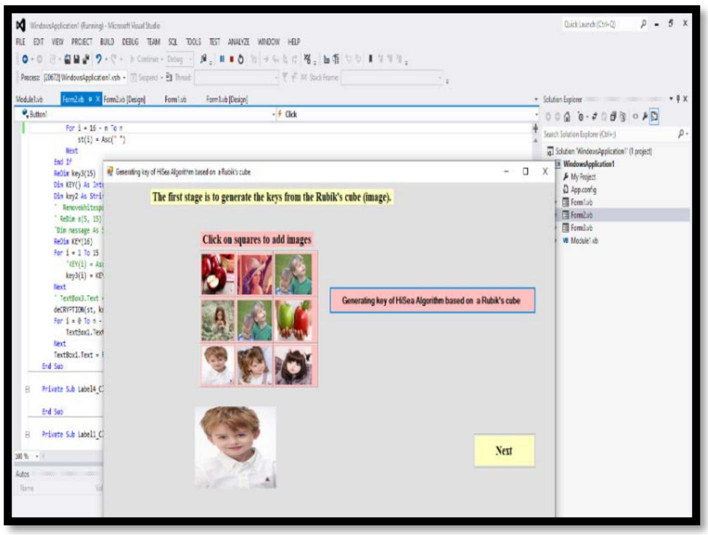
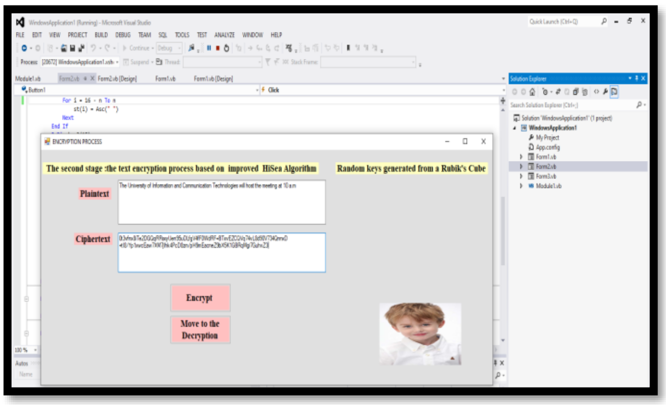

Generating key of HiSea Algorithm based on a Rubik's cube
The first stage is to generate the keys from the Rubik's cube (image).
Click on squares to add images
Generating key of HiSea Algorithm based on a Rubik's cube
Next

Figure 7: Generate the Hi sec algorithm key from the Rubik's Cube.

The original message “The University of Information and Communication Technologies will host the meeting at 10 a.m. “is encrypted in the second stage. which is 0t3vfnx8iTe2DGQqRRsxyUen95uDLfgV4fF0WdRF+BTxvEZCQVq74vL8d90V734QnnxD+tI8/Yp1xwcEaw

7XM7jIhk4PcD8zm/pH9mEacneZ9bX5K1GBRqMgi7GuhvZ3", according to figure 8.

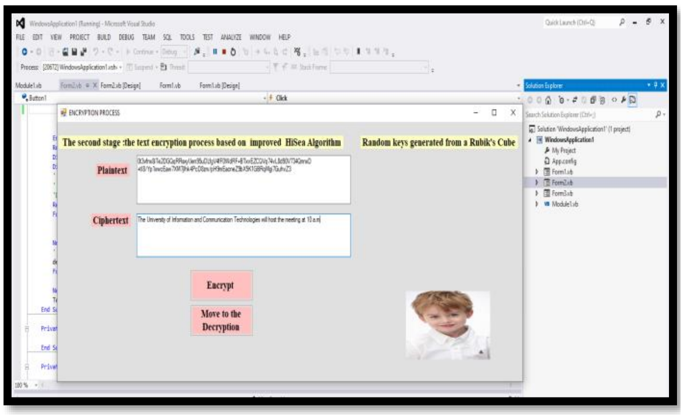

WindowsApplication1 (Running) - Microsoft Visual Studio
ENCRYPTION PROCESS
The second stage :the text encryption process based on improved HiSea Algorithm
Random keys generated from a Rubik's Cube
Encrypt
Move to the Decryption

Plaintext: The University of Information and Communication Technologies will host the meeting at 10 a.m

Ciphertext: [illegible]
+t8/Yp1wvoEaw7XM7jIhk4PcD8zm/pH9mEacneZ9bX5K1GBRqMgi7GuhvZ3

Figure 8. Encrypted Message using Improving Hi Sec Algorithm.

The final stage involves retrieving the original message using the improved Hi sec algorithm by the receiving side using the encrypted message as depicted in figure 9.


WindowsApplication1 (Running) - Microsoft Visual Studio
ENCRYPTION PROCESS
The second stage :the text encryption process based on improved HiSea Algorithm
Random keys generated from a Rubik's Cube
Encrypt
Move to the Decryption

Plaintext: [illegible]
+t8/Yp1wvoEaw7XM7jIhk4PcD8zm/pH9mEacneZ9bX5K1GBRqMgi7GuhvZ3

Ciphertext: The University of Information and Communication Technologies will host the meeting at 10 a.m

Figure 9. Original Message Based on Improving Hi Sec Algorithm.

V. Statistical Tests:

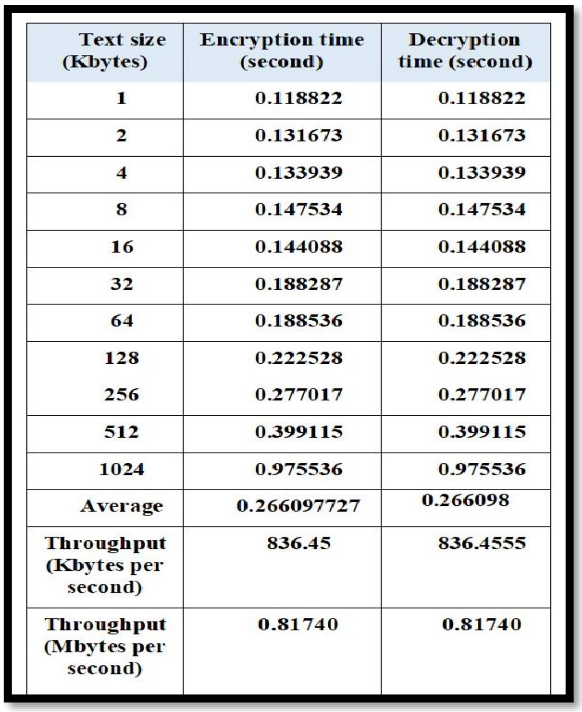
On the generated keys, I ran several statistical tests to gauge the security of our key creation process. We tested the security and randomness of cryptographic keys using the NIST Statistical Test Suite, a widely used tool. Our findings revealed that the keys produced by our technology passed all of the NIST tests, proving that they are secure and random. Our method outperformed other Rubik's Cube-based key generation methods in terms of unpredictability and security, according to the same statistical tests we used to compare our method's security to those of the competitors. Overall, our technology presents a promising strategy for producing Hi Sea algorithm secure keys based on Rubik's Cube image. As a standard evaluation tool for evaluating the randomness of generated sequences, the NIST Statistical Test Suit was created [14,15]. The 15 tests of the tool assess the significance of the p-value for each run. The sequence passes the randomness test if the p-value is ≥ to 0.01 percent. The unpredictability of various algorithms is assessed using the test suit [16–17]. Table 2 contains 15 tests of the suggested algorithm.

Table 2: Randomness NIST Evaluations of the Suggested Algorithm

No	Type of Test	P-Value	Result
1	Frequency Test	0.55526	Pass
2	Frequency Test within a Block	0.61227	Pass
3	Run Test	0.57346	Pass
4	Longest Run of Ones in a Block	0139138	Pass
5	Rank Test	0.93910	Pass
6	Discrete Fourier Transform Test	0.87122	Pass
7	Non-Overlapping Template Matching Test	0.98608	Pass
8	Overlapping Template Matching Test	0.24449	Pass
9	Maurer's Universal Statistical test	0.85863	Pass
10	Linear Complexity Test	0.91968	Pass
11	Serial test	0.41422	Pass
12	Approximate Entropy Test	0.99999	Pass
13	Cumulative Sums Test	0.48897	Pass
14	Random Excursions Variant Test:	0.72564	Pass
15	Random Excursions Test:	0.95024	Pass

Table 3 displays the experimental findings that were attained for text files of various sizes, together with the encryption and decryption times.

Table 3: Text files Cryptography Results



Text size (Kbytes)	Encryption time (second)	Decryption time (second)
1	0.118822	0.118822
2	0.131673	0.131673
4	0.133939	0.133939
8	0.147534	0.147534
16	0.144088	0.144088
32	0.188287	0.188287
64	0.188536	0.188536
128	0.222528	0.222528
256	0.277017	0.277017
512	0.399115	0.399115
1024	0.975536	0.975536
Average	0.266097727	0.266098
Throughput (Kbytes per second)	836.45	836.4555
Throughput (Mbytes per second)	0.81740	0.81740

Table 4 displays the observed experimental findings. Numerous short communications were also taken and encrypted-decrypted.

Table 4: Short Messages (Small Text Files) Cryptography Results

Algorithm / file size	Hi sec Algorithm	Improving Hi sec Algorithm
11	0.0141896522	0.0178414336
15	0.0145186051	0.0184446787
17	0.0145896145	0.0155178889
26	0.0176051896	0.0116941465
30	0.01189451760	0.0133444187
40	0.01618942605	0.0134582244
50	0.0164276023	0.0135668768
60	0.0118966035	0.0134768923
70	0.0163342633	0.0154684633
80	0.0105165266	0.0144687783
90	0.016460448	0.0144988763
100	0.0164605342	0.0144989265
Average	0.014756888	0.0146037814

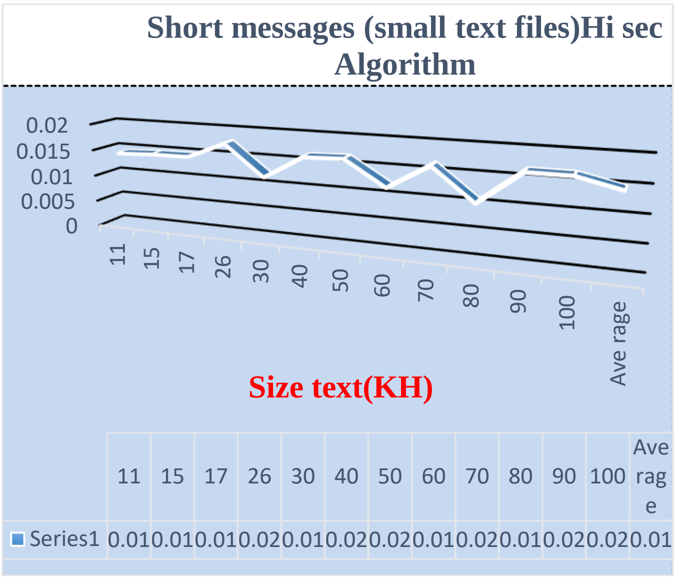
The results are provided in Table 4 and were applied using the same brief messages and the same processes. Each of the customary techniques was employed to encrypt and decrypt the same huge text files. Table 5 presents the experimental results that were obtained.

Table 5: Results of text file encryption using standard techniques

file size for text (K bytes)	timing of encryption (second)	
	Hisses Algorithm	Improving Hi sec Algorithm
1024	33.222 7	Yes
512	41.336 4	Yes
256	22.4438	Yes
128	6.4373	Yes
64	2.2786	Yes
32	1.4471	Yes
16	0.7533	Yes
8	0.3344	Yes
4	0.1556	yes
2	0.1772	yes
1	0.0528	No
Average	3.786678	
Throughput (K bytes per second)	25.8884	

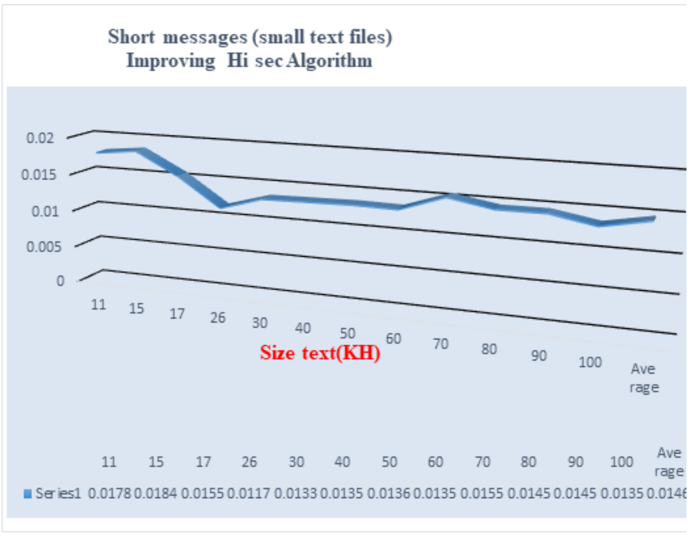
VI. Results Analysis

By offering outstanding values during the encryption and decryption phases, the suggested technique satisfies the quality criteria, as can be seen from the findings presented in tables 2, 3, and Figures 10 and 11 com comp the compare encryption times of the implemented methods with the proposed picture for short messages cryptography (see tables 3, 4, and 5).



	11	15	17	26	30	40	50	60	70	80	90	100	Ave rag e
■ Series1	0.01	0.01	0.01	0.02	0.01	0.02	0.02	0.01	0.02	0.01	0.02	0.02	0.01

Figure 10. Short messages (small text files) Hisses Algorithm.



	11	15	17	26	30	40	50	60	70	80	90	100	Ave rage
■ Series1	0.0178	0.0184	0.0155	0.0117	0.0133	0.0135	0.0136	0.0135	0.0155	0.0145	0.0145	0.0135	0.0146

Figure 11. Short messages (small text files) Improving Hi sec Algorithm

The suggested strategy considerably boosts data cryptography throughput, as shown in Figures 10 and 11. When the size of the text file grows, the encryption-decryption times for the current techniques quickly rise (see table 6).

VII.CONCLUSION

By increasing the efficiency and throughput of the encryption-decryption process, the proposed approach quickly outperformed the previously used traditional methods of data cryptography. This key is to be created by a Rubik's Cube, making the hacking process impossible. The newly created approach provides an even greater level of data security and privacy through the use of a Rubik's Cube. Using a new approach to image encryption that generates a random key from a Rubik's cube, exhibits encryption speed, and can perform good encryption and resist analytical attacks. Short communications and text files of any size can be easily protected using the suggested technique. A Rubik's Cube can be used with asymmetric cryptography techniques in the future. The suggested approach can also be modified to incorporate more media, such as audio, video, etc.

VIII. ACKNOWLEDGMENT

The author would like to thank Informatics Institute for Postgraduate Studies, Iraqi Commission for Computers and Informatics (https://iips.edu.iq/) Baghdad - Iraq for its support in the present work

IX. REFERENCES

[1]. Prateek Rawat, Ritwik Mishra, and Aseem Upadhyay," TEXT ENCRYPTION BY RUBIK'S CUBE USING SPATIAL STEGANOGRAPHY", ISST Journal of Mathematics & Computing System, Vol. 7 No. 2, (July - December 2016), p.p. 53-59. ISSN No. 0976-9048 © Intellectuals Society for Socio-Techno Welfare

[2]. Priyan, K. and Vishal, P," Design and Implement Dynamic Key Generation to Enhance DES Algorithm", International Journal for Research in Applied Science & Engineering Technology, Volume 4 Issue VII, July 2016, IC Value: 13.98, ISSN:2321-9653. https://www.ijraset.com/fileserve.php?FID=5278

[3]. V. Magesh Babu, T. Shankar Ganesh, K. Ramraj," A Comparative Analysis on Encryption and Decryption Algorithms", Int J Sci Res Publ 4(12)

(ISSN:2250-3153) (2018) http://www.ijsrp.org/research-paper-1214.php?rp=P36346.

[4]. T. de Castella, "The people who are still addicted to the Rubik's Cube," BBC News Magazine. BBC. com. Retrieved, vol. 28, 2014.

[5]. M. Helmy, E.-S. M. El-Rabaie, I. M. Eldokany, and F. E. A. El-Samie, "3-D Image Encryption Based on Rubik's Cube and RC6 Algorithm," 3D Research, vol. 8, no. 4, p. 38, 2017. https://link.springer.com/article/10.1007/s13319-017-0145-8

[6]. K. Loukhaoukha, J.-Y. Chouinard, and A. Berdai, "A secure image encryption algorithm based on Rubik's cube principle," Journal of Electrical and Computer Engineering, vol. 2012, p. 7, 2012. https://doi.org/10.1155/2012/173931

[7]. K. Abitha and P. K. Bharathan, "Secure Communication Based on Rubik's Cube Algorithm and ChaoticBaker Map," Procedia Technology, vol. 24, pp. 782-789, 2016. https://doi.org/10.1016/j.protcy.2016.05.089.

[8]. S. Kilaru, Y. Kanukuntla, A. Firdouse, and M. Bushra, "effective and key sensitive security algorithm for an image processing using robust Rubik encryption and decryption process," the University of Birmingham, ISSN (Print), vol. 2, pp. 2278-8948, 2013. http://www.irdindia.in/journal_ijaeee/pdf/vol2_iss5/17.pdf

[9]. Zhu, H., Dai, L., Liu, Y., & Wu, L, "A three-dimensional bit-level image encryption algorithm with Rubik's cube method', Mathematics and Computers in Simulation, 185, 754-770. (2021). DOI: 10.1016/j.matcom.2021.02.009

[10]. A. Bogdanov and K. Shibutani, "Generalized Feistel networks revisited," Designs, Codes and Cryptography, vol. 66, pp. 75–97 (2013). https://link.springer.com/article/10.1007/s10623-012-9660-z

[11]. S. Panasenko and S. Smagin, "Lightweight Cryptography: Underlying Principles and Approaches," International Journal of Computer Theory and Engineering, vol. vol 3., 2011. DOI: 10.7763/IJCTE. 2011.V3.360

[12]. Alyaa Ghanim Sulaiman1, Sufyan Salim Mahmood AlDabbagh2," Modified 128-EEA2 Algorithm by Using HISEC Lightweight Block Cipher Algorithm with Improving the Security and Cost Factors", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 10, No. 1, April 2018, pp. 337~342, ISSN: 2502-4752, DOI: 10.11591/ideas.v10.i1.pp337-342.

[13]. SufyanSalimMahmoodAldabbaghb, Mustafa Ali Abuzaraidac, AlyaaGhanimd Khalid Abdulkareem Al-Enezie," Improving Highest Security Lightweight block cipher (HISEC) Algorithm Using Key Dependent S-box", Turkish Journal of Computer and Mathematics Education Vol.12 No.3(2021), 3544-3549, https://doi.org/10.17762/turcomat.v12i3.1630.

[14]. N. Sp- and N. Sp-, The NIST Statistical Test Suite, (2021) 1–6.

[15]. M. Sýs and Z. Říha, Faster randomness testing with the NIST statistical test suite, Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 8804 (2014) 272–284. https://doi.org/10.1007/978-3-319- 12060-7_18

[16]. E. Yavuz, R. Yazıcı, M. C. Kasapbaşı, and E. Yamaç, A chaos-based image encryption algorithm with simple logical functions, Comput. Electr. Eng., 54 (2016) 471–483. https://doi.org/10.1016/j.compeleceng.2015.11.008

[17]. Z. Hua, F. Jin, B. Xu, and H. Huang, 2D Logistic-Sine-coupling map for image encryption, Signal Processing, 149 (2018) 148–161. https://doi.org/10.1016/j.sigpro.2018.03.010

Cite this article as :

Israa Shihab Ahmed , "Text Encryption Using Improving Key of Hisses Algorithm Using Rubik's Cube", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 10 Issue 2, pp. 218-226, March-April 2023. Available at doi : https://doi.org/10.32628/IJSRSET2310222

Journal URL : https://ijsrset.com/IJSRSET2310222