Proceedings of the 3rd International Conference on Mathematical Physics and Computational Simulation
DOI: 10.54254/2753-8818/100/2025.21702

Theoretical Analysis of AES Encryption Optimization Based on Quaternions and Rotation Matrices

Yuchen Wu[1,a,*]

[1]*Mathematics and Applied Mathematics, Jiangsu University, Zhenjiang, Jiangsu, 212013, China*
a. 1970482859@qq.com
**corresponding author*

Abstract: As a commonly used symmetric encryption method, AES (Advanced Encryption Standard) still has some steps that cannot fully ensure the effectiveness of encryption, and Symmetric Key Encryption has the risk of key leakage in common. This paper mainly researches an optimization model of AES encryption method, which is based on the original AES encryption method. This model adds another password in the SubByte and MixColumns step, in order to ensure the effectiveness of encryption. The optimization model is achieved with programs on python. Experimental data shows that the encryption process is too regular due to the methodologies of the model itself: the location of those bytes after mapping can be separated into some sets, and the transforms of this model can't exchange the mapping of those bytes from different sets. After analyzing and optimizing the model, another improved model is worked out, which can simply confuse the bytes with their locations. The experiments show that the next model has a better encrypting effect than the first one.

Keywords: Quaternion, Rotation Matrix, Advanced Encryption Standard, 4-dimensional Rubik's Cube

1. Introduction

In the existing AES (Advanced Encryption Standard) encryption principle, each round of encryption can be divided into four main parts of the operation: SubBytes, ShiftRows, MixColumns and AddRoundKey, which uses the key only in the last step AddRoundKey, that means, the security from password is only guaranteed by this step, the other steps are mainly contributing to obfuscate the encrypted bytes corresponding to the plaintext, but the deciphering of those steps don't rely on any password. This means that the encryption effect of the first three steps can be removed by finite steps of processing on computer, and the security of the encryption itself should not be guaranteed by the complexity of the encryption method, so it can be determined that there is still room for optimization here.

Inspired by the high dimensional Rubik's Cube, in this paper, a new way of encryption is found, which gives an algorithm that relies on symmetric keys to generate encrypted matrices by matrix rotation. This is capable of upgrading the security of the existing encryption principle of AES. Like AES maps the whole byte into a 2-dimensional ByteMatrix, the optimization proposed in this paper will map the whole byte to a 4-dimensional fourth-order matrix, and use the idea of rotating (similar to the Rubik's Cube) to define a new "Rotate" and "Exchange" encryption, and then determine the encryption sequence through the password to ensure its security. The encryption sequence depends

DOI: 10.54254/2753-8818/100/2025.21702

on a password to ensure its security and effectiveness. This part of the research is implemented and debugged on Python. Through the model proposed in this paper, it will be able to improve the original encryption effect and confidentiality level of AES with more keys, ensure the effectiveness of other encryption steps, and reduce the risk of information leakage.

2. Decipherability analysis of the original SubBytes

2.1. Mathematical evidence of commutative reversibility and feasibility of inverse commutative summation

The AES encryption algorithm establishes a bijection between bytes during SubByte by means of a fixed S-box, which in turn disrupts the correspondence of the bytes to their original positions [1]. A is the byte matrix before SubByte, and S is the one-to-one correspondence transformation performed by the S-box:

$$S(A) \mapsto B \tag{1}$$

B represents the swapped byte matrix. However, since the S-box is publicly and uniquely determined, it is straightforward to find a fixed inverse mapping S^{-1} in both normal decryption and unconventional deciphering:

$$S^{-1}(B) \mapsto A \tag{2}$$

For a state byte matrix, if the result of a SubByte step is subjected to the inverse SubByte operation S^{-1}, the result of that SubByte encryption step will be recovered. This is a threat to the overall security of AES encryption, as will be demonstrated below.

2.2. Ciphertexts handling and the possibility of corresponding key failure

As a symmetric encryption, all the processing in AES encryption is reversible. Except for AddRoundKey, all the steps in AES can be cracked directly without a password, that is, the corresponding position of the relationship can be worked out before the password is hacked. There's a round of encryption as an example:

$$A_i \oplus K_i = B_i, C \circ R \circ S(B_i) = P_i, P_i \oplus W_i = A_{i+1} \tag{3}$$

Here is the operation performed for one single round of encryption, K_i is the round key (unknown to the decryptor), R is the ShiftRows transformation, C is the MixColumns transformation, and W_i is the new round key obtained by the processing of K, which is set temporarily:

$$W_i = W(K_i) \tag{4}$$

Then it follows from the fact that their inverse operations all exist and are unique:

$$A_{i+1} \oplus W(K_i) = P_i, S^{-1} \circ R^{-1} \circ C^{-1}(P_i) = B_i, B_i \oplus K_i = A_i \tag{5}$$

The equation in the previous line can be verified by the principle of superposition of inverse maps and the self-reflexivity of the Exclusive OR. However, if K_i is in an unknown state during the deciphering process, the above steps should be easily reordered:

$$Q_i = ((C \circ R \circ S)^{-1}(A_{i+1})$$

$$f(A_{i+1}, K_i) = (C \circ R \circ S)^{-1}(W(K_i)) \oplus Q_i) \oplus K_i = A_i \tag{6}$$

DOI: 10.54254/2753-8818/100/2025.21702

This method provides an operation to reset the position of the disrupted bytes without a password, which can eliminate the effect of all the disruptions in the encryption, in order to prepare for the decryption step that requires a key. However, the remaining part of the key decryption will also face the problem of "the corresponding position of the key does not match the ciphertext", because each round of key stacking is for the encrypted byte matrix after the disruption, so it can not be directly substituted. However, the keys can also be pretreated like those byte matrix do in the opposite order of encryption, and ultimately a transposition function can be given out, after whose pretreatment the encrypted byte matrix only needs to add those keys in the correct order:

$$\mathrm{Re}_i(K_i) = (C \circ R \circ S)^{1-i}(K_i), \mathrm{Re}_i(W_i) = (C \circ R \circ S)^{1-i}(W(K_i)) \tag{7}$$

After substituting the key, this function will solve a pretreated key

Sequence, which can decrypt the plaintext by perform XOR summation operations with the byte matrix in the inverted order:

$$T(A_n) = (C \circ R \circ S)^{n-1}(A_n)$$

$$g(A_n, \{K_1, \dots, K_n\}) = e_1\big(W(K_1)\big) \oplus \mathrm{Re}_1(K_1) = A_1$$

$$= T(A_n) \oplus \mathrm{Re}_n(W(K_n)) \oplus \mathrm{Re}_n(K_n) \oplus \dots \oplus R \tag{8}$$

To the matrix after transposed, the security is at the same level as there's only AddRoundKey step in the encryption. And all those keys in different rounds are combined with the different-or-sum operation, which means the number of rounds can't increase the security. So it is envisioned that the step of SubBytes can also be encrypted in order to protect the degree of security.

2.3. Effect of security enhancement by adding SubBytes key

The reason that the method above can create an inverse mapping for those encryption steps in AES and pretreat the encryption results for batch decryption is that the transforms used for its disruption are same and unique, and if each round is disrupted with a different S-Box or other method that relies on another key, it will make the new inverse computation extremely complex and will also need to rely on a new key. Let the hall permutation transformation of round i be $Ro_i(A_i)$:

$$Ro_i(A_i) = C \circ R \circ S_i(A_i) \tag{9}$$

S_i is a key-dependent SubByte transformation, then the previous decryption formula should be changed to:

$$T'(A_n) = (\prod_{i=1}^{n} Ro_i)(A_n), \mathrm{Re}'_1(K_1) = K_1, \mathrm{Re}'_i = (C \circ R \circ S_i)^{-1} \circ \mathrm{Re}'_{i-1}$$

$$g'(A_n, \{K_1, \dots, K_n\}) = A_1$$

$$= T'(A_n) \oplus \mathrm{Re}'_n(W(K_n)) \oplus \mathrm{Re}'_n(K_n) \oplus \dots \oplus \mathrm{Re}'_1(W(K_1)) \oplus \mathrm{Re}'_1(K_1) \tag{10}$$

And each round of Ro_i and Re_i here depends on the transformation key. This leads to a new model in which it is not possible to preprocess the matrices to eliminate the permutation transforms without the key. And the reason for using fixed permutation transforms for encryption in AES is that complex 16*16 permutation bijection is difficult to generate in bulk. In the following section, we will give an idea of permutation transforms based on Rubik's Cube disrupted by algorithm, which can realize key dependent permutation encryption transformation.

DOI: 10.54254/2753-8818/100/2025.21702

3. Feasibility proof of 4-dimensional dot matrix corresponding to byte

3.1. 4-dimensional rotation effect

The rotating transform of a 16*16 byte matrix leads to a very similar result because of the simple structure of a 2-dimensional matrix; and a 3-dimensional matrix is very difficult to establish correspondence with the byte due to its quantitative peculiarities ($3^k \neq 256, k \in Z$). Compared to a 16*16 2-dimensional matrix or a 3-dimensional matrix structure, a 4-dimensional fourth-order matrix can both correspond one-to-one with 256 bytes and provide sufficient complexity for rotation-based disruption. An attempt is made to construct a 4-dimensional 4*4*4*4 node-square matrix to correspond to 256 different bytes.

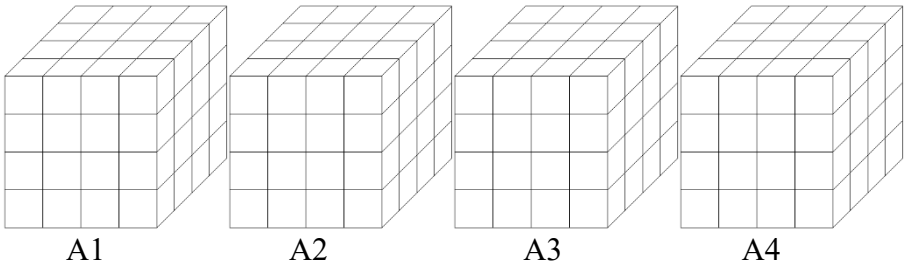

Figure 1: Schematic diagram of a 4-dimensional fourth-order matrix

As shown in Fig. 1, the 4th order 4-dimensional matrix A can be interpreted as a vector consisting of four 4th order 3-dimensional matrices A_1, A_2, A_3, A_4, with an extra dimension orthogonal to all the original directions on top of the 3-dimensional matrix. Define the new center-symmetric coordinates for each element of this high-dimensional matrix, such as the element $a = A_1(-1,1,2)$ in $A1$ (the center-symmetric coordinates are used here for the convenience of rotational transformation afterwards), and then define the coordinates corresponding to A_1, A_2, A_3, A_4 in the new dimension to be -2,-1,1,2, respectively, so that there is $a = A_1(-1,1,2,-2)$ in the 4-dimensional matrix.

The following passage will prove that the original definition of matrix rotation for any 3-dimensional A_i are still applicable on 4-dimensional matrix A.

The t rotational transformation, rotating a 2-dimensional matrix (like the kth layer of matrix A_i) for t times, marked as $Ro^*(k,t)$, can be expressed as follows: left-multiply the plane coordinates (i,j) of all the elements $A_i(i,j,k)$ of the kth layer by the rotate matrix $Rotate^*(t)$:

$$Rotate^*(t) = \begin{bmatrix} \cos\ t\pi/2 & -\sin\ t\pi/2 \\ \sin\ t\pi/2 & \cos\ t\pi/2 \end{bmatrix} \tag{11}$$

This transformation still works for any 2-dimensional matrix component in a 4-dimensional matrix:

$$Ro(k,l,t): A(i,j,k,l) \mapsto A(i\cos\frac{t\pi}{2} - j\sin\frac{t\pi}{2}, i\sin\frac{t\pi}{2} + j\cos\frac{t\pi}{2}, k, l) \tag{12}$$

It is worth to notice that there are 2 dimensions orthogonal to the rotated plane in the 4-dimensional space, i.e., the rotary shaft in the 4-dimensional space is 2-dimensional, so at least 2 direction parameters are required to determine the rotary shaft for one rotation, and thus the rotational transformation requires 2 positional parameters in each direction to determine the rotated plane, and an integer parameter to determine the time of rotation.

In addition, rotations with a 4th dimension are not included in the rotation transformations in 3D space (e.g., $Ro(i,k,t)$, which does not exist as a mapping in 3D space), but can be constructed according to the above method. In fact, from the above statement "the rotary shaft has 2 dimensions", it can be determined that there are a totally $C_2^4 = 6$ rotary shaft directions in 4-dimensional space [2].

DOI: 10.54254/2753-8818/100/2025.21702

3.2. Proof of feasibility of expressing rotated (encrypted) sequences via multiple quaternion array

It is known that a rotation in one axis can be represented in three dimensions by a single quaternion. Similarly, rotation in four dimensions needs to be limited to two axes, and rotation for a single layer needs to be limited to the coordinates of that layer in both axes, otherwise the entire 4-dimensional matrix would be rotated. Here 4 orthogonal unit quaternions are needed to represent the directions, so a new axis l is added; accordingly, the constants used to represent the isometric scaling are not necessary. A more complete definition of the rotation transformation $Rotate(d1, d2, h1, h2, t)$ is given here.

Let $d1, d2$ be the two axial directions, $h1, h2$ be the coordinates of the plane in the axial directions, and t be the number of rotations. With *i,j,k,l* representing the four directions, the plane generated by the two directions orthogonal to both $d1$ and $d2$ is rotated. At this point the forward direction of the rotation should be considered: the rotation by quaternions should satisfy: $d1^{*}d2 = -d2^{*}d1 (d1 \neq d2)$, which defines the sequentiality of the sequence $i \to j \to k \to i$ and maps the inverse order multiplication to be the reverse result [3-4]. It is also considered to continue following this definition in the 4-dimensional case: Firstly, construct a structure in 3-dimensional space, each vertex means one sign of *i,j,k,l*, then with the spacial relationship of this structure, the mapping order of quaternion arrays can be defined:

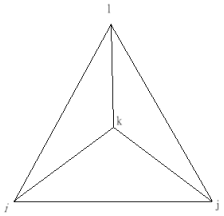

Figure 2: Schematic diagram of the correspondence between axial direction and rotation direction

As in Fig. 2, *Rotate(i,j)* is the rotation in the axial direction of *(i,j)*, while *(k,l)* is the order of the rotation direction. *Rotate(i,j)* maps the coordinates of the original *(k,l)* plane to the new coordinates by left-multiplicating the rotation matrix to the vector $(h_k, h_l)^T$, while the coordinates of the axial direction in the plane of *(i,j)* remains unchanged. Where (h_k, h_l) is the original coordinates of *(k,l)* , i.e. the nodes on the rotating plane. The following mapping relations (see Table 1) are established with this order rule:

Table 1: Correspondence between the axial direction (left) and the rotational direction (right) of the Rotate function.

Rotate(i,j)	(k,l)	Rotate(j,k)	(i,l)	Rotate(k,l)	(i,j)	Rotate(l,i)	(k,j)
Rotate(i,k)	(l,j)	Rotate(j,l)	(k,i)	Rotate(k,i)	(j,l)	Rotate(l,j)	(i,k)
Rotate(i,l)	(j,k)	Rotate(j,i)	(l,k)	Rotate(k,j)	(l,i)	Rotate(l,k)	(j,i)

A rotation in four dimensions can then be uniquely expressed with two ordered axes and a rotating angle parameter. Since the rotation that will be discussed later is a rotation of a hypercube layer (with two dimensions), it is necessary to specify the coordinates of the two axes to determine the plane in which the object of rotation is located, named as *h1* and *h2*. The function $Rotate(d1, d2, h1, h2, t)$ can be used to represent the axes of *d1* and *d2*, and the plane consisting of the nodes with coordinates *h1* in the direction of *d1* and *h2* in the direction of *d2* can be rotated by t*90° in the order of the

DOI: 10.54254/2753-8818/100/2025.21702

corresponding directions (with *d1,d2* as the axis). This rotational transformation acts on the corresponding nodes, and the coordinates of the mapped nodes still form a 4-dimensional hypercube.

3.3. Feasibility proof of applying rotations determined by quaternion array to the SubBytes encryption

In order to apply the 4-dimensional cubic rotation method defined above to cryptography, it is necessary to ensure that the transform is a bijection, which means, there is no loss of information in the matrix after the rotation. The fact that 4-dimensional rotation does not result in loss of information can be directly deduced by the fact that rotation on a single plane does not result in loss of information, since a 4-dimensional rotation essentially changes the byte state of only one planar component. Whereas plane rotations are invertible transformations, any rotation has a unique inverse mapping, which means that for any image of that mapping there exists a unique preimage, so the mapping does not lead to the overwriting of valid information. It is also known that any plane of the hypercube is centrosymmetric about the shaft of that plane, and a rotation with an angle of $k\pi/2, (k \in \mathrm{Z})$ must still fall on the original coordinates, which means the information before the rotation will not be lost even after the rotation of a single plane, thus proving it.

4. Add key model design (methods for generating encryption based on keys)

4.1. Rotated Byte Encryption Model

In order to realize the encryption model described previously, the correspondence between bytes and spatial locations needs to be constructed. Compared to building the spatial structure, assigning coordinates to each byte can determine the rotation transformation more Intuitively, because the rotation matrix left-multiplication of the byte coordinates can achieve the purpose, and this can also avoid the problems encountered in the byte transformation, such as the overwriting of valid information.

4.2. Program implementation and analysis

Table 2: Rotation encryption effect

encrypted rounds	1	2	3	4	5	6	7	8
Maximum value of invariant bytes	133	78	50	34	29	21	18	19
invariant byte minimum	71	22	11	9	6	6	4	5
Invariant byte count average	97	47	28	20	15	13	11	11

In order to observe the encryption effect, those data shown in Table 2 are the number of nodes whose corresponding coordinates are the same as the initial state through the algorithm after 16 random rotational transformations in each round , and the data obtained after 100 experiments are (the third row is the result after rounding)[5]. The reason why we do not choose to follow the time complexity and avalanche effect of encryption to measure the efficiency and security of this encryption method is that the object of study here is only one step transformation in the encryption process, not a complete encryption method, and a single-step encryption transformation can not be directly compared with a complete encryption method [6].

The number of unbroken bytes after encryption decreases rapidly in the earlier rounds, yet this change is no longer significant after 4 rounds. This is mainly due to a property of the model itself.

DOI: 10.54254/2753-8818/100/2025.21702

When an element with 4-dimensional coordinates is rotated, the range of new coordinates it will be mapped to is restricted, and elements whose coordinates have a class of features will still only be mapped to coordinates with the same features after any rotational encryption. In this way, those bytes can be roughly categorized into five sets of elements according to their coordinate features:

$$D_0: \{x \mid x = (k1, k2, k3, k4), |k1| + |k2| + |k3| + |k4| = 4\}$$

$$D_1: \{x \mid x = (k1, k2, k3, k4), |k1| + |k2| + |k3| + |k4| = 5\}$$

$$D_2: \{x \mid x = (k1, k2, k3, k4), |k1| + |k2| + |k3| + |k4| = 6\}$$

$$D_3: \{x \mid x = (k1, k2, k3, k4), |k1| + |k2| + |k3| + |k4| = 7\}$$

$$D_4: \{x \mid x = (k1, k2, k3, k4), |k1| + |k2| + |k3| + |k4| = 8\} \quad (13)$$

The number of elements in each set can be easily derived from the previous hypercube legend:

$$card(D_0) = 16, card(D_1) = 64, card(D_2) = 96, card(D_3) = 64, card(D_4) = 16$$

The coordinates that belonged to D_i before the rotation will only fall into D_i after the rotation, which provides degree of predictability to the rotation transformation itself, and means that the degree of disruption after the rotation is limited.

4.3. Cryptographic improvement optimization and comparison

Therefore, we consider adding another transformation: Exchange, which exchanges all the coordinates of the bytes in two parallel different 2-dimensional layers. It can be easily concluded that the transformation is still reversible by the one-to-one correspondence of the coordinates themselves, and the transformation can disrupt the nodes belonging to different sets of D_0, D_1, D_2, D_3, D_4. The degree of encryption is further improved by randomly adding the swap transform to each round of encryption (See Table 3).

Table 3: Encryption effect after adding swapping (8 random transformations per round)

Number of encrypted rounds after joining the exchange	1	2	3	4	5	6	7	8
Maximum value of invariant bytes	163	100	57	36	19	12	9	7
invariant byte minimum	91	31	8	3	0	0	0	0
Invariant byte count average	122	57	26	14	7	4	3	1

The data shown in Table 3 comes from the encryption effect after adding the exchange transformation, and only 8 random rotations or exchanges (after rounding the third row) were performed in each round in order for the data to significantly reflect the encryption ability of the model. It can be seen that by the above method can significantly improve the encryption efficiency, then a better model is obtained.

5. Conclusion

In this paper, based on the original AES encryption, the security of the Subbytes encryption link is analyzed, and a new encryption model is further designed and improved to optimize its encryption effect. The experimental results show that the optimized encryption results have higher unpredictability and security, and can provide better protection for encrypted objects. In this paper, the research didn't show the effect of the complete optimized AES encryption and algorithmic time

DOI: 10.54254/2753-8818/100/2025.21702

complexity change of the improved AES encryption obtained after replacing the original Subbytes link with the new encryption model. Also, the research didn't analyze the avalanche effect by comparing the strength of the two. In the next research, there are still many improvements that can be made, such as replacing the Subbytes link with this encryption optimization model, making the complete encryption program obtained into a product, and analyzing the performance of its memory usage, arithmetic power consumption, and encryption security through the above methods.

References

[1] *Jingmei Liu, Baodian Wei, Xiangguo Cheng and Xinmei Wang, "An AES S-box to increase complexity and cryptographic analysis," 19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers), Taipei, Taiwan, 2005, pp. 724-728 vol.1, doi: 10.1109/AINA.2005.84.*

[2] *Hashimoto, Hideya. "Hypersurfaces in 4-dimensional Euclidean space." Czechoslovak Mathematical Journal 40.2 (1990): 315-324.*

[3] *Siddiqui, S., Samad, M.A. & Ismoiljonovich, F.D. One dimensional quaternion linear canonical transform in probability theory. SIViP 18, 9419–9430 (2024). https://doi.org/10.1007/s11760-024-03556-9*

[4] *Mukundan, R. (2012). Quaternions. In: Advanced Methods in Computer Graphics. Springer, London. https://doi.org/10.1007/978-1-4471-2340-8_5*

[5] *WYC0016/test_of_encryption: A model used to optimize SubBytes step in AES encryption,GitHub - WYC0016/test_of_encryption: A model used to optimize SubBytes step in AES encryption*

[6] *A. K. Mandal, C. Parakash and A. Tiwari, "Performance evaluation of cryptographic algorithms: DES and AES," 2012 IEEE Students' Conference on Electrical, Electronics and Computer Science, Bhopal, India, 2012, pp. 1-5, doi: 10.1109/SCEECS.2012.6184991.*