2023 IEEE 5th International Conference on Civil Aviation Safety and Information Technology (ICCASIT) | 979-8-3503-1060-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCASIT58768.2023.10351719

Research on Enhanced AES Algorithm Based on Key Operations

Shilin Liu
Network Information Security Laboratory
Yanbian University
Yanji, 133002, China
2022050082@ybu.edu.cn

Yongzhen Li
Network Information Security Laboratory
Yanbian University
Yanji, 133002, China
lyz2008@ybu.edu.cn

Zhexue Jin*
Network Information Security Laboratory
Yanbian University
Yanji, 133002, China
jinzhexue@ybu.edu.cn

***Abstract*—AES is recognized as one of the most secure symmetric encryption algorithms and has been widely used in the field of information security.It has the advantages of simplicity, efficiency, symmetry, modularity, etc., but there are many attacks against AES, such as differential attack, linear attack, side channel attack, etc., which requires us to further study the methods to enhance the security of AES algorithm. This paper presents an enhanced AES algorithm based on key operation.The traditional SubBytes operation in AES replacing the current state matrix with values obtained from the S-box. and the ShiftRows is that every row of the state matrix is rotated to the left according to the fixed bit, while the improved method mentioned in this paper is to associate the SubBytes and ShiftRows with the key in the calculation process. After the improvement of SubBytes, the current state matrix and the KV value calculated by the wheel key matrix are XOR operation to get a new state matrix, and the new state matrix is used to replace the S-box.The ShiftRows improvement is to dynamically shift the state matrix by comparing the KV values obtained from the calculation of the wheel key. The efficiency of the algorithm will be reduced slightly due to the introduction of key-related operations in the process of this improvement scheme, but it is worthwhile to improve the security of the algorithm on the basis of the small decrease in efficiency. Experiments show that the improved AES algorithm has better avalanche effect than the traditional AES algorithm.**

Keywords—AES algorithm, shift rows, byte substitution, key dependency, avalanche effect

I. Introduction

In the current era of big data, the security protection of data has become increasingly important. With the continuous development of computer computing power, people are paying more attention to network information security and have a stronger awareness of protecting their personal information. In order to better safeguard information security, it is necessary to enhance the construction of network security [1].

Compared to public-key encryption, symmetric encryption algorithms have advantages such as fast encryption speed and shorter key length[14]. Currently, widely used symmetric ciphers can be classified into two categories: stream ciphers and block ciphers. The representative stream cipher is RC4[15], while block ciphers include DES[16], IDEA[17], and AES, among others.AES, as a classical symmetric encryption algorithm, was established as the Advanced Encryption Standard in 2001[2]. It is known for its high encryption efficiency and security. The AES algorithm mainly consists of several operations, including SubBytes, ShiftRows, MixColumn, AddRound Key, and key expansion. Among these steps, byte substitution is the only nonlinear transformation in the AES algorithm.Therefore, byte substitution is a key operation to improve the performance of the AES algorithm[3]. As the most widely used and highly secure symmetric encryption algorithm, AES has faced various attacks such as differential attacks [4], side-channel attacks[5], cache timing attacks [6], and power analysis attacks[7]. Cryptanalysts have conducted strength analysis on AES, revealing that with the continuous enhancement of computational power, eight rounds of AES can be successfully brute-forced out of ten rounds[8]. This urges the exploration of further enhancing the security of the AES algorithm. In this paper, we focus on strengthening the security of the byte substitution and shift rows processes in AES. Taking the key length of 128 bits as an example, the initial key generates 10 circular sub-keys through the key expansion process[9,10]. This paper proposes proposed improvement approach adopts the concept of key dependency[11], making each round's byte substitution and shift rows processes rely on the corresponding subkey. As a result, any change in a single bit of the key significantly affects the resulting ciphertext, providing better diffusion and avalanche effect during the encryption process[12]. This key-dependent enhancement provides a means to strengthen the security of the algorithm while minimizing any adverse effects on operational efficiency.

II. AES Encryption Algorithm

A. AES encryption and decryption process

The AES encryption algorithm, based on the key length of 128 bits, 192 bits, and 256 bits, corresponds to the number of encryption/decryption iterations as 10, 12, and 14 rounds, respectively. In this article, the 128-bit key length is used as an example. The AES encryption process consists of four components:SubBytes, ShiftRows, MixColumn, and AddRound Key[2].

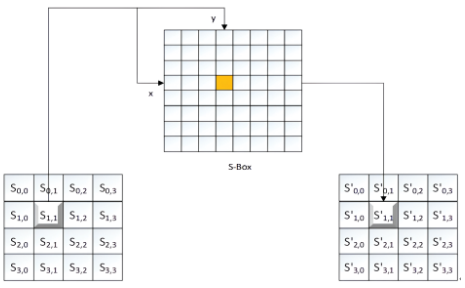

Fig. 1. SubBytes.

979-8-3503-1060-3/23/$31.00 ©2023 IEEE

SubBytes is the process of replacing each byte in the state matrix with a corresponding byte defined by the AES S-box [13], as shown in “Fig 1”.

ShiftRows rearranging the state matrix according to the shifting operation depicted in “Fig 2”.

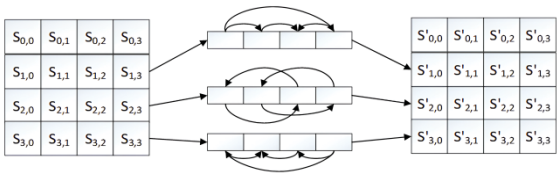

Fig. 2. ShiftRows.

MixColumn operation is performed by left-multiplying the state matrix with a predefined mixing matrix, as depicted in “Fig 3”.

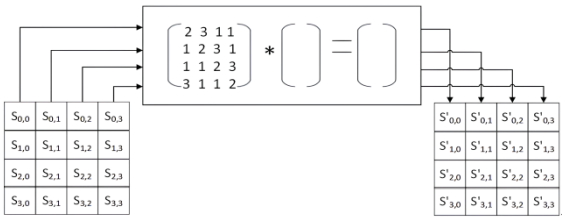

Fig. 3. MixColumn.

AddRound Key operation is performed by applying a bitwise XOR operation between the state matrix for each round and its corresponding round key, as illustrated in “Fig 4”.

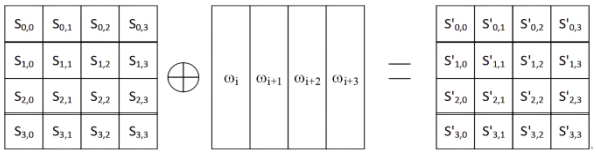

Fig. 4. AddRound Key.

AES is a symmetric encryption algorithm, and its decryption process involves reversing the operations applied during the encryption process. The detailed flow of AES encryption and decryption is presented in “Fig 5”.

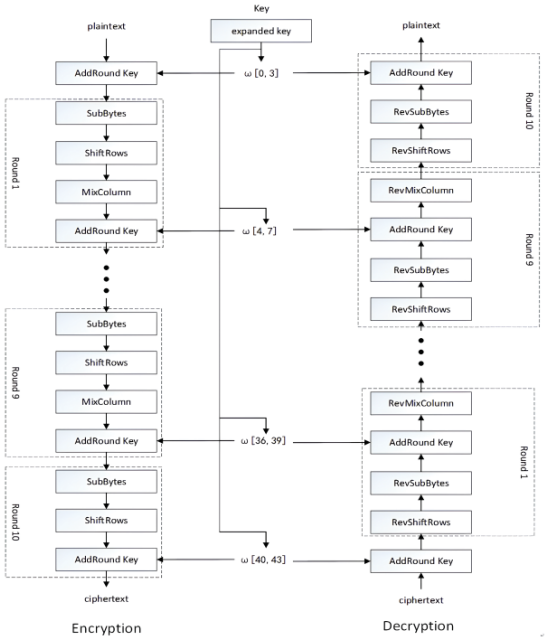

Fig. 5. AES encryption and decryption process diagram.

B. AES key extension

To generate subkeys for AES, a key expansion operation is performed on a 128-bit seed key. The seed key is evenly divided into four 32-bit words: ω_0, ω_1, ω_2, and ω_3, with each word consisting of four bytes. This 128-bit keys are used as the round keys for the initial round calculations[9,10].

“Fig 6” illustrates the process of generating round keys. The subkey generation starts with the calculation of ω_4 using ω_0 and ω_3. Next, ω_1 and ω_4 are used to compute ω_5. Similarly, ω_2 and ω_5 are utilized to derive ω_6, and ω_3 and ω_6 are employed to obtain ω_7. The generated subkeys ω_4, ω_5, ω_6, and ω_7 serve as the round keys for the second round. This calculation process continues iteratively to calculate the round keys for subsequent rounds, resulting in a total of 10 round keys.

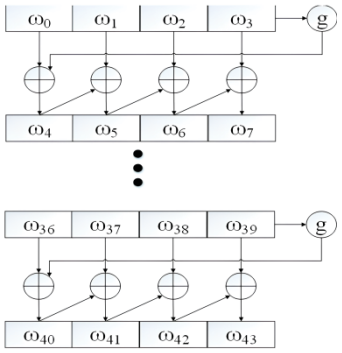

Fig. 6. AES key extention algorithm process.

III. Proposed Improvement Method

The AES algorithm itself possesses excellent modularity and symmetry, which allows designers to modify a specific part of the algorithm's structure without negatively impacting the normal operation of other parts.This paper presents two improvement strategies for the AES algorithm, specifically targeting the byte substitution process and the row shifting process. The proposed enhancements in this study involve making the computation of row shifting and byte substitution dependent on the key, thereby increasing the algorithm's resistance to attacks.

A. Improvements to SubBytes

Before the improvement, SubBytes process involves directly substituting the elements of the current state matrix with those from the S-box. The improved approach involves performing a computation between the current state matrix S and the round subkey matrix K to obtain a new state matrix S'.The specific computation involves XOR operations between the four 8-bit values in each row of the 4x4 subkey matrix. This results in four 8-bit values, namely KV_0, KV_1, KV_2, and KV_3. Then, each of these four values is XOR with the corresponding elements in the current state matrix's rows. For example, KV_0 is XOR with each element of the first row

of the current state matrix, and this process is repeated for the other rows accordingly. By doing so, a new state matrix S' is obtained. The new state matrix S' is then subjected to SubBytes using the S-box. The detailed calculation process is presented below.

The current state matrix S and the subkey matrix K are represented as formula is shown in (1) .

$$S=\begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33} \end{bmatrix} \quad K=\begin{bmatrix} K_{00} & K_{01} & K_{02} & K_{03} \\ K_{10} & K_{11} & K_{12} & K_{13} \\ K_{20} & K_{21} & K_{22} & K_{23} \\ K_{30} & K_{31} & K_{32} & K_{33} \end{bmatrix} \tag{1}$$

The process of calculating the four KV values obtained through each round's subkey computation is shown in formula (2)-(5).

$$KV_0 = K_{00} \oplus K_{01} \oplus K_{02} \oplus K_{03} \tag{2}$$

$$KV_1 = K_{10} \oplus K_{11} \oplus K_{12} \oplus K_{13} \tag{3}$$

$$KV_2 = K_{20} \oplus K_{21} \oplus K_{22} \oplus K_{23} \tag{4}$$

$$KV_3 = K_{30} \oplus K_{31} \oplus K_{32} \oplus K_{33} \tag{5}$$

The new state matrix S' is calculated by using the four KV obtained from each round of subkey calculation and the current state matrix S. The specific calculation process is shown in formula (6) as follows.

$$S'=\begin{bmatrix} S_{00} \oplus KV_0 & S_{01} \oplus KV_0 & S_{02} \oplus KV_0 & S_{03} \oplus KV_0 \\ S_{10} \oplus KV_1 & S_{11} \oplus KV_1 & S_{12} \oplus KV_1 & S_{13} \oplus KV_1 \\ S_{20} \oplus KV_2 & S_{21} \oplus KV_2 & S_{22} \oplus KV_2 & S_{23} \oplus KV_2 \\ S_{30} \oplus KV_3 & S_{31} \oplus KV_3 & S_{32} \oplus KV_3 & S_{33} \oplus KV_3 \end{bmatrix} \tag{6}$$

It is important to note that the decryption process of the improved SubBytes is reverse to the encryption process. Firstly, the inverse S-box is used for reverse SubBytes to obtain S'.Then, the initial state matrix S is restored by XOR KV_0, KV_1, KV_2, and KV_3 with the corresponding elements in accordance with the key-dependent calculation method. This ensures that the algorithm can decrypt the initial plaintext from the ciphertext. “Fig 7” provides a detailed comparison of the internal processes of the improved byte substitution in the encryption and decryption processes.

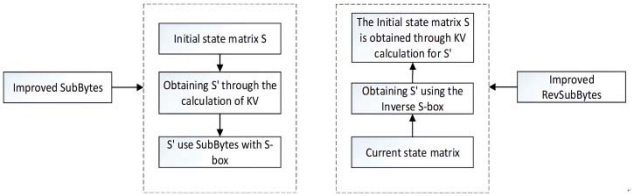

Fig. 7. Improved SubBytes scheme internal processes.

B. Improvements to ShiftRows

In this paper, a key-dependent dynamic ShiftRows improvement scheme is proposed. The calculation of the subkeys in each round yields four KV values, which determine the shift values for each row of the current state matrix. The specific computation process involves comparing the four KV values obtained from the current round's subkeys. In the encryption process, the row with the largest KV value is shifted left by three positions, the row with the second-largest KV value is shifted left by two positions, the row with the third-largest KV value is shifted left by one position, and the row with the smallest KV value remains unchanged. In the case of two equal KV values, the row with the larger row number is considered greater than the row with the smaller row number. For example, if the comparison result of the four KV values is $KV_2 > KV_1 > KV_0 > KV_3$, it means that the first row of the state matrix is shifted left by one position, the second row is shifted left by two positions, the third row is shifted left by three positions, and the fourth row remains unchanged.

The following is an example of dynamic row shifting, where the current state matrix S and the subkey matrix K for the current round are shown as formula (7).

$$S=\begin{bmatrix} 3C & 4A & 04 & 58 \\ 64 & 05 & C4 & 13 \\ 80 & 37 & F3 & 46 \\ 4B & 65 & 26 & D8 \end{bmatrix} \quad K=\begin{bmatrix} 47 & 6B & 82 & 37 \\ 55 & A4 & D4 & 46 \\ 89 & 85 & 05 & F6 \\ 4A & 45 & 26 & C8 \end{bmatrix} \tag{7}$$

The four KV values, KV_0, KV_1, KV_2, and KV_3, are computed using the method shown in formula (8)-(11).

$$KV_0 = 47 \oplus 6B \oplus 82 \oplus 37 = 99 \tag{8}$$

$$KV_1 = 55 \oplus A4 \oplus D4 \oplus 46 = 63 \tag{9}$$

$$KV_2 = 89 \oplus 85 \oplus 05 \oplus F6 = EF \tag{10}$$

$$KV_3 = 4A \oplus 45 \oplus 26 \oplus C8 = E1 \tag{11}$$

Based on the calculations performed, it can be inferred that $KV_2 > KV_3 > KV_0 > KV_1$. By comparing the KV values, we can determine the shift values for each row of the state matrix. Further details are provided in Table I.

TABLE I. KV VALUE AND SHIFT VALUE

Serial number	KV	Hexadecimal	Decimal	Shift value	Line
1	KV_0	99	153	1	0
2	KV_1	63	99	0	1
3	KV_2	EF	239	3	2
4	KV_3	E1	225	2	3

Using the calculated KV values from each round's subkey computation, the comparison of these four KV values determines the shift values for the state matrix. The dynamic row shifting of the current state matrix is then conducted based on the computed shift values obtained from Table I. Formula (12) illustrates the original state matrix S before shifting and the resulting state matrix S' after shifting.

$$S=\begin{bmatrix} 3C & 4A & 04 & 58 \\ 64 & 05 & C4 & 13 \\ 80 & 37 & F3 & 46 \\ 4B & 65 & 26 & D8 \end{bmatrix} \quad S'=\begin{bmatrix} 4A & 04 & 58 & 3C \\ 64 & 05 & C4 & 13 \\ 46 & 80 & 37 & F3 \\ 26 & D8 & 4B & 65 \end{bmatrix} \tag{12}$$

The decryption process entails reversing the dynamic row shifting. KV values are computed from each round's subkey. AES is a symmetric encryption algorithm, whereby the same key is used for both encryption and decryption. As a result, the computed KV values remain unchanged. By comparing the sizes of the four KV values, the corresponding rows of the

state matrix can be right-shifted according to the calculated shift values.

IV. Experimental Results and Analysis

A. Efficiency analysis

As shown in Table II, it presents a comparison between the encryption and decryption time of traditional AES and the improved AES for different file sizes. To eliminate randomness, I performed encryption and decryption on 100 identical files using the same key, and calculated the average encryption and decryption times for each file. In the table, "ET" represents encryption time, and "DT" represents decryption time.

TABLE II. Comparison of Encryption and Decryption Time

File size	Traditional AES(s)	Improved AES(s)
16 Bytes	ET: 0.000977	ET: 0.002346
	DT: 0.001723	DT: 0.003645
32 Bytes	ET: 0.001726	ET: 0.004548
	DT: 0.003269	DT: 0.005346
64 Bytes	ET: 0.002678	ET: 0.009045
	DT: 0.005049	DT: 0.011316
128 Bytes	ET: 0.003946	ET: 0.015458
	DT: 0.008449	DT: 0.025738

From Table II, we can clearly observe that in the improved algorithm, after adding key-dependent patterns to byte substitution and row shifting, the encryption and decryption time slightly increase compared to traditional AES for plaintext of various byte lengths. While efficiency experiences a slight decrease, the security of the algorithm is significantly enhanced. This improvement provides better resistance against attack techniques such as differential attacks and linear attacks.

B. Avalanche Effect Test

Avalanche effect refers to the phenomenon where flipping a single bit in the plaintext or key of an encryption algorithm results in a significant number of differing bits in the generated ciphertext compared to the ciphertext obtained before the flip. When any bit in the input plaintext or key is flipped, each bit in the output has a 50% probability of changing. In other words, changing a single bit in the plaintext or key leads to substantial variations in the resulting ciphertext, which demonstrates the reliability of the security of the encryption algorithm. Therefore, the avalanche effect is commonly employed to assess the effectiveness of encryption algorithms.

According to the principle of the avalanche effect, when a single bit in the plaintext or key is flipped, there is a 50% probability of each bit in the output ciphertext changing. In an ideal situation, when the plaintext or key flips a bit, the ratio of the number of different bits between the two ciphertexts (before and after the flips) to the total number of bits in the ciphertext should be close to 0.5. In this experiment, points with ratios less than 0.4 or greater than 0.6 are defined as fault points, and the traditional AES and improved AES are compared.

This experiment randomly generates a 64-byte plaintext and a 16-byte key for the AES encryption algorithm. The individual bits in the key are randomly flipped and the same plaintext is encrypted using the original key and the modified key. Compare the corresponding bits in the generated ciphertext and calculate the ratio of different bits to the total number of bits in the ciphertext. In an ideal situation, this ratio should be 0.5, and to avoid chance, we repeated 1000 rounds of the experiment and plotted the avalanche effect values of both traditional AES and improved AES encryption algorithms on a scatter plot. This experiment aims to demonstrate the effect of the proposed modification on avalanche effect.

As shown in “Fig 8”, in the case of traditional AES, where a random bit in the key is flipped for the 1000 rounds of avalanche effect experiment, there were 12 points with ratios greater than 0.6 and 18 points with ratios less than 0.4. In other words, a total of 30 faulty points were observed.

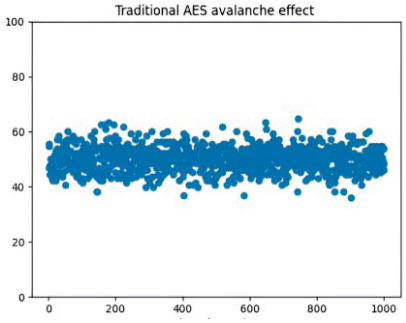

Fig. 8. Traditional AES flips keys with an avalanche effect of one bit.

As depicted in “Fig 9”, for the improved AES, where a random bit in the key is flipped for the 1000 rounds of avalanche effect experiment, there were 10 points with ratios greater than 0.6 and 10 points with ratios less than 0.4. Consequently, a total of 20 faulty points were observed.

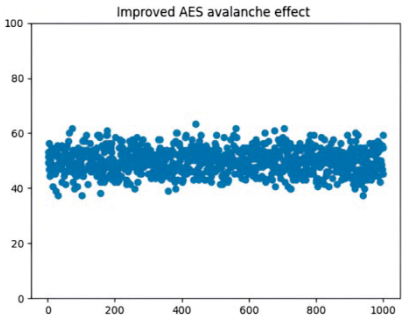

Fig. 9. Improved avalanche effect of AES flipping keys by one bit.

To reduce randomness, a total of five sets of avalanche effects were tested against conventional AES and improved AES, each consisting of 1,000 rounds. The results of each experiment, including the number of faulty points, average faulty points, and the probability of faulty points, are summarized in Table III.

TABLE III. Flip Key Avalanche Effect Comparison

Algorithm name	1	2	3	4	5	Average failure	Failure probability
Traditional AES	18	24	26	30	30	26	2.6%
Improved AES	24	23	22	22	20	22	2.2%

Randomly generate 64 bytes of plaintext and 16 bytes of key. Random bits in plaintext are flipped, and 1000 rounds of avalanche effect experiments are carried out on the traditional AES encryption algorithm and the improved AES encryption algorithm. The 1000 avalanche effect values are plotted on a scatter plot.

As depicted in “Fig 10”, in the case of traditional AES, where a random bit in the plaintext is flipped for the 1000

rounds of avalanche effect experiment, there were 15 points with ratios greater than 0.6 and 12 points with ratios less than 0.4. In other words, a total of 27 faulty points were observed.

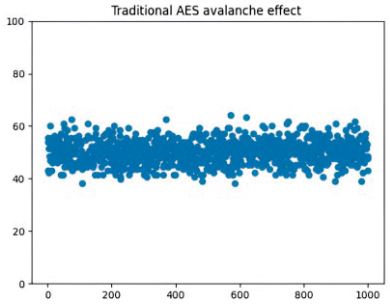

Fig. 10. Traditional AES flips plaintext by one bit of avalanche effect.

As shown in "Fig 11", for the improved AES, where a random bit in the plaintext is flipped for the 1000 rounds of avalanche effect experiment, there were 7 points with ratios greater than 0.6 and 13 points with ratios less than 0.4. Consequently, a total of 20 faulty points were observed.

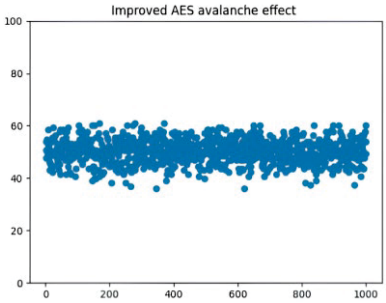

Fig. 11. Improved AES flipping plaintext with one bit of avalanche effect.

To reduce randomness, a total of five sets of avalanche effects were tested against conventional AES and improved AES, each consisting of 1,000 rounds. The results of each experiment, including the number of faulty points, average faulty points, and the probability of faulty points, are summarized in Table IV.

TABLE IV. FLIPPED PLAINTEXT AVALANCHE EFFECT COMPARISON

Algorithm name	1	2	3	4	5	Average failure	Failure probability
Traditional AES	16	28	28	28	27	24	2.4%
Improved AES	23	21	27	25	20	23	2.3%

By comparing the experimental results above, it is evident that the improved AES algorithm exhibits a lower probability of faulty points compared to the traditional AES algorithm. This indicates that the improved AES algorithm demonstrates better avalanche effect and diffusion properties.

V. CONCLUSION

In this paper, we propose a new scheme based on key operations by improving the SubBytes and ShiftRows operations in AES. This scheme utilizes the computation of four different KV values from the subkeys and introduces them into the computation process of SubBytes and ShiftRows. The improved scheme introduces KV values related to the subkeys during the encryption process, enhancing the nonlinearity of the AES algorithm.By incorporating the KV values computed from the subkeys into the SubBytes operation, the resistance against attacks such as statistical analysis, linear cryptanalysis, and differential cryptanalysis is strengthened. The use of these four KV values to determine the ShiftRows operation improves the confusion effect of the operation and enhances the confidentiality of the algorithm. Experimental results demonstrate that the improved encryption algorithm exhibits enhanced avalanche effect and diffusion properties, reducing the probability of faulty points caused by the avalanche effect. In conclusion, the security of the algorithm is successfully improved by improving the SubBytes and ShiftRows operations in AES.

REFERENCES

[1] Zhang Huanguo, Wu Fusheng, Wang Houzhen, Wang Zhangyi. A review of security verification analysis of cryptographic protocol code execution[J].Journal of Computer Science, 2018, 41(02): 288-308.

[2] Wang Y. AES Encryption and Decryption Algorithm and Its Security Analysis. Network Security Technology and Applications, 2022(09): 33-35.

[3] Yuan F, Jiang JJ, Yang Y, Ou HW, Wang MJ. The cryptographic function counting problem with inverse function affine equivalence over a finite fieldF_(p~n)[J].Journal of Computer Science, 2019, 42(05): 1126-1136.

[4] Blondeau C, Leander G, Nyberg K. Differential-linear cryptanalysis revisited. J Cryptol, 2017, 30: 859‑888.

[5] Raphael Spreitzer, Veelasha Moonsamy, Thomas Korak, and Stefan Mangard. Systematic classification of sidechannel attacks: A case study for mobile devices. IEEE Communications Surveys & Tutorials, 2018, 20(1): 465– 488.

[6] C. Rebeiro, P. H. Nguyen, D. Mukhopadhyay and A. Poschmann. "Formalizing the Effect of Feistel Cipher Structures on Differential Cache Attacks," in IEEE Transactions on Information Forensics and Security, vol. 8, no. 8, pp. 1274-1279, Aug. 2013.

[7] Zhang H, Liu G, Zhang X. Research on Energy Analysis Attack Based on AES Algorithm. Yangtze River Information and Communication. 2021, 34(07): 21-24.

[8] Zodpe, H.; Sapkal, A. An Efficient AES Implementation using FPGA with Enhanced Security Features. J.King Saud Univ. Eng. Sci. 2020, 32: 115-122.

[9] Liu X. An Improved AES Algorithm and Its Performance Analysis. Yangtze River Information and Communication. 2021, 34(11): 27-29.

[10] Liu Y, Li Q. Research on AES Algorithm and Improvement of Its Key Expansion Algorithm. Modern Electronics Technology. 2016, 39(10): 5-8.

[11] O. C. Abikoye, A. D. Haruna, A. Abubakar, N. O. Akande, and E. O. Asani, "Modified advanced encryption standard algorithm for information security," Symmetry, 2019, 11: 12.

[12] Deng Y, Xie T, Shi H. Study on the Avalanche Effect Characteristics of AES Algorithm's Key [C] China Electronics Education Society. Proceedings of the 2011 Conference of the Higher Education Division of China Electronics Education Society. 2011: 84-89.

[13] A. Seghier, J. Li and D.Z. Sun, "Advanced encryption standard based on key dependent S-Box cube", IET Information Security, 2019, 13(6): 552-558.

[14] Stoyanov B, Nedzhibov G. Symmetric key encryption based on rotation-translation equation.Symmetry, 2020, 12(1): 73.

[15] Gupta SS, Chattopadhyay A, Sinha K, Maitra S, Sinha BP. High-performance hardware implementation for RC4 stream cipher. IEEE Trans Comput, 2012 ,62(4): 730–743.

[16] Zhang J, Zhu L.Analysis and Implementation of DES Encryption Algorithm [J]. Software Guide, 2007(03): 95-97.

[17] Basu S. International data encryption algorithm (IDEA)–a typical illustration. J Glob Res Comput, 2011, 2(7): 116–118.