

# Research on Data Encryption Standard Based on AES Algorithm in Internet of Things Environment

Na Su<sup>1</sup>, Yi Zhang<sup>2,\*</sup>, Mingyue Li<sup>1</sup>

<sup>1</sup> College of Information Science and Engineering, Guilin University of Technology  
Jiangan road No.12, Guilin, China

<sup>2</sup> College of Information Science and Engineering, Guilin University of Technology  
Jiangan road No.12, Guilin, China

mail: 471144957@qq.com, E-mail: 997732295@qq.com, \*Corresponding author: zywait@glut.edu.cn

**Abstract-** The Internet of Things makes it possible to exchange information between people, things and things, people and things, and makes information security more and more important. However, there are few standards in the field of information security of the Internet of Things. Therefore, based on the characteristics and application requirements of the Internet of Things environment, this paper optimizes the Advanced Encryption Standard (AES) and builds the data encryption standard DESI(Data Encryption Standard in IoT) in the Internet of Things environment. . The analysis shows that DESI has better security and is therefore suitable for encrypting data in the IoT environment. **Keywords:** Internet of Things, encryption standard, AES, DESI, information security.

**Keywords-** Internet of Things, encryption standard, AES, information security

## I. INTRODUCTION

The Internet of Things is widely used in China's logistics, power, medical, environmental protection and transportation [1]. The application of the Internet of Things has effectively reduced operating costs and improved economic efficiency. At the moment, security issues are another huge challenge facing the rapid development of the Internet of Things [2]. Encryption technology can provide effective protection for data security of the Internet of Things.

## II. RESEARCH ON AES ENCRYPTION ALGORITHM OF INTERNET OF THINGS

Due to the shortness of DES's key space and its relatively small packet size, the National Institute of Standards and Technology began to publicly collect Advanced Encryption Standard (AES) from the world in 1997. After more than three years, the Rijndael algorithm designed by two Belgian cryptographers Daemen and Rijmen eventually won from many candidates and became the advanced encryption standard. AES itself has the advantages of high speed, high security, and easy hardware and software implementation [3].

The AES algorithm is basically identical to the Rijndael algorithm. The difference between the two is that the packet length and key length are different. The Rijndael algorithm has a variable packet length and key length, which can be set to an integer multiple of 32 bits, ranging from 128 to 256 bits. The AES algorithm fixes the

packet length to 128 bits, and the key length can only be one of 128, 192, and 256 bits [4]. The number of rounds of the AES algorithm is related to the key length. The dependencies are shown in Table I.

III. TABLE I RELATIONSHIP BETWEEN KEY LENGTH AND NUMBER OF ROUNDS IN AES

Key length (bits)	Number of rounds
128	10
192	12
256	14

Figure 1 illustrates the encryption process for the 10-round AES algorithm

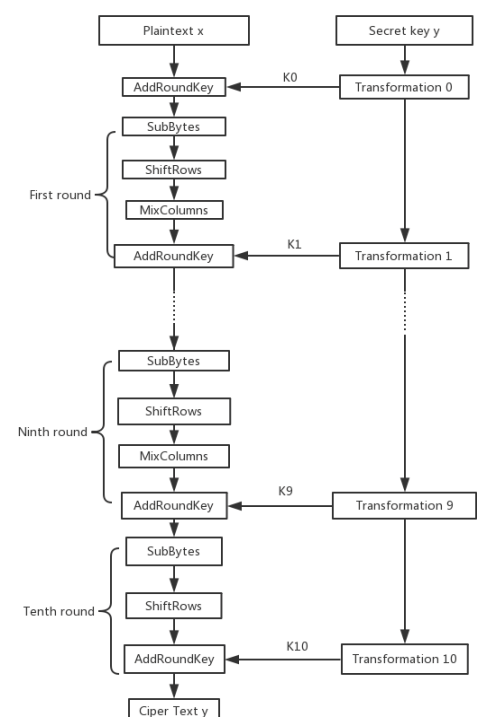


Figure 1 AES encryption process

The internal structure of the AES algorithm consists of four basic steps [5], namely SubBytes, ShiftRows, MixColumns, and AddRoundKey. Each of these links

operates on the entire 128-bit plaintext packet, also known as the algorithm state. The state of the algorithm consists of 16 bytes, which is usually represented as a matrix of 4 rows and 4 columns.

#### A . Byte substitution.

Byte substitution is the only non-linear operation in the AES algorithm, and it is also an important part of ensuring the security of the AES algorithm. In byte substitution, AES uses 16 identical S-boxes, each of which is an 8-bit input, an 8-bit output, and these S-boxes work in parallel. With byte substitution, each byte in the algorithm state is non-linearly replaced with a new byte. The generation of the S box in the AES algorithm is performed in two steps:

① Convert a byte to its multiplicative inverse in  $GF(2^8)$ . It is worth noting that the multiplication inverse of the 0 element does not exist, but the multiplication of the AES definition 0 element is inversely mapped to itself.

②The affine transformation is performed on the result of the first step, and the affine transformation is defined as  $b=f(a)$ , that is, each byte is first multiplied by a constant matrix, and an 8-bit hexadecimal constant vector  $\{63\}$  is added, such as a formula. As shown in formula (1)

$$\begin{pmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} a_7 \\ a_6 \\ a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad (1)$$

#### B . Line shift.

The row shift is an operation of cyclically shifting each row of the algorithm state according to different displacement amounts. The first line of the algorithm state does not need to be changed, the second line of the algorithm state is rotated right by three bytes, the third line

$$\begin{bmatrix} a00 & a01 & a02 & a03 \\ a10 & a11 & a12 & a13 \\ a20 & a21 & a22 & a23 \\ a30 & a31 & a32 & a33 \end{bmatrix} \oplus \begin{bmatrix} k00 & k01 & k02 & k03 \\ k10 & k11 & k12 & k13 \\ k20 & k21 & k22 & k23 \\ k30 & k31 & k32 & k33 \end{bmatrix} = \begin{bmatrix} a00 \oplus k00 & a01 \oplus k01 & a02 \oplus k02 & a03 \oplus k03 \\ a10 \oplus k10 & a11 \oplus k11 & a12 \oplus k12 & a13 \oplus k13 \\ a20 \oplus k20 & a21 \oplus k21 & a22 \oplus k22 & a23 \oplus k23 \\ a30 \oplus k30 & a31 \oplus k31 & a32 \oplus k32 & a33 \oplus k33 \end{bmatrix}$$

Figure 4 Round key plus transformation

#### E . Key expansion algorithm.

The key expansion algorithm refers to how to generate sub-keys of each round through the initial key of the AES algorithm. It is worth noting that the number of subkeys is not equal to the number of rounds, but it is necessary to add 1 to the number of rounds. The initial key of AES is optional 128-bit, 192-bit, and 256-bit, and the corresponding rounds are 10, 12, and 14 rounds

is rotated right by two bytes, and the fourth line is rotated right by one byte. As shown in Figure 2. The effect of the row shift transform enhances the degree of diffusion of the AES algorithm, which in turn ensures the security of the algorithm.

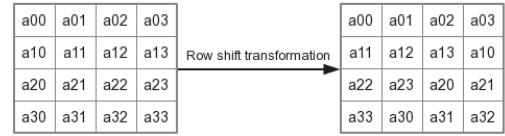


Figure 2 Row shift transformation

#### C . Column mixing.

A column blending transformation is an operation of separately performing a blending transformation on each column of an algorithm state. Think of each column of the algorithm state as a polynomial with a coefficient on  $GF(2^8)$ , multiply by a fixed polynomial  $c(x)$ , and then a modular polynomial  $(x^4 + 1)$ , where  $c(x)$  is defined as:

$$c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (2)$$

Then each byte in the algorithm state is mapped to another value, and this value is related to the 4 bytes of the column. Column blending transformations can be represented by matrix multiplication, as shown in Figure 3.

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a00 & a01 & a02 & a03 \\ a10 & a11 & a12 & a13 \\ a20 & a21 & a22 & a23 \\ a30 & a31 & a32 & a33 \end{bmatrix} = \begin{bmatrix} a'_{00} & a'_{01} & a'_{02} & a'_{03} \\ a'_{10} & a'_{11} & a'_{12} & a'_{13} \\ a'_{20} & a'_{21} & a'_{22} & a'_{23} \\ a'_{30} & a'_{31} & a'_{32} & a'_{33} \end{bmatrix}$$

Figure 3 Column hybrid transformation

#### D . Round key addition.

Round key addition is a bitwise XOR operation of a 16-byte algorithm state matrix and a 16-byte subkey. The subkey is obtained by the key expansion by the initial key. Figure 4 illustrates the flow of round key addition.

respectively. The key expansion algorithm corresponding to the three key length AESs has certain similarities. In order to avoid the generality, the key expansion algorithm will be described by taking AES with a key length of 128 bits as an example.

In the AES algorithm, the subkey is calculated recursively, that is, if the subkey  $w_i$  is to be generated,

the precondition is that the known subkey  $w_{i-1}$  is required. In addition, the basic unit of key spreading by the AES algorithm is a 32-bit word. The key expansion algorithm takes 4 words as input, and the output of the algorithm is a one-dimensional array of 44 words.

First, the initial key of AES is used as the first 4 key words of the extended key array. Thereafter, the newly generated four key words are added to the remaining portion of the extended key array each time. Within the extended key array, the generation of each new key word  $w_i$  depends on  $w_{i-1}$  and  $w_{i-4}$ . At the same time, the algorithm uses a more complex function  $g$  to calculate the key word whose array index is a multiple of 4. Figure 5 illustrates how the extended key is calculated.

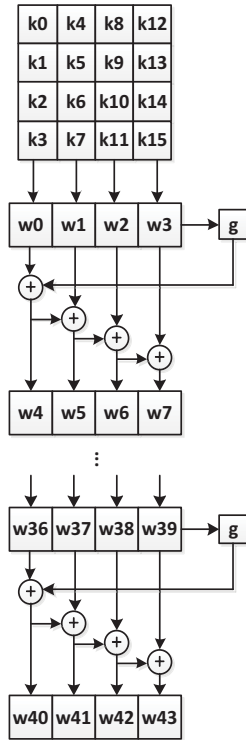


Figure 5 Key Expansion Algorithm

As can be seen from the figure, the key word  $w_i$  is calculated as:

- ① When  $i \bmod 4 \neq 0$ ,  $w_i = w_{i-4} \oplus w_{i-1}$ ;
- ② When  $i \bmod 4 = 0$ ,  $w_i = w_{i-4} \oplus g(w_{i-1})$ ;

The function  $g()$  is a non-linear function of 4-byte input and 4-byte output. In the  $g()$  function, first, the four input bytes are sequentially rotated left by one byte; then, the four S-boxes are used to perform byte substitution on the shifted four bytes; finally, the byte is replaced. The result is XORed with the round constant  $Rcon[j]$ . The round constant is a 32-bit word, and the other three bytes are 0 except that the leftmost byte is not 0. So,

the key word is XORed with the round constant, which is actually the XOR of the first byte to the left of the round constant. It should be noted that the round constants of each round are different, specifically defined as:  $Rcon[j] = (Rc[j], 0, 0, 0)$ , where  $Rc[1] = 1$ ,  $Rc[j] = Rc[j-1] * 2$  (this multiplication is defined on  $GF(2^8)$ ). Table II is a hexadecimal representation of the ten round constants used in the AES algorithm.

TABLE II ROUND CONSTANT

j	1	2	3	4	5	6	7	8	9	10
$Rc[j]$	01	02	04	08	10	20	40	80	1B	36

#### IV. AES-BASED ENCRYPTION STANDARD DESIGN IN THE INTERNET OF THINGS

When determining the number of rounds of the encryption algorithm, it is usually necessary to consider the resistance of the algorithm to the shortcut attack. The reason is very simple. The shortcut attack is more efficient than the exhaustive search key. However, in an IoT environment where computing resources and storage resources are relatively short, the number of rounds of the encryption algorithm is not as good as possible. Therefore, when designing the encryption standard DESI in the Internet of Things, this paper considers to properly reduce the number of rounds of the algorithm, and improve the execution efficiency of the algorithm under the premise of ensuring the security of the algorithm. By consulting the literature, it can be known that for the AES algorithm with a key length of 128 bits, the shortcut attack is only effective for versions with a round of 6 rounds or less, and more than 6 rounds can withstand known shortcut attacks [6]. Therefore, this paper decided to design the number of rounds of the encryption algorithm to 7 rounds, of which 6 rounds can be used to resist shortcut attacks, and the remaining 1 round is used to ensure the security of the algorithm [7].

In order to verify the rationality of the 7-round algorithm, an initial key with a length of 128 bits is randomly selected. In this set of initial keys, one and only one is different. The AES key expansion algorithm performs 10 rounds of key expansion on the initial key, then observes and analyzes the change of the extended key, and obtains the bit difference of each round key according to the generated round key. Among them, the bit difference refers to the sum of the number of different bits in the corresponding position in each group of round keys [8]. According to the principle of the encryption algorithm, it can be known that the round key will be applied to each round of round key addition transformation, so the larger the bit difference of the round key, the better the diffusion effect of the key after the key expansion, the algorithm The higher the security.

The random initial key is shown in Table 3.1. In the

initial set of keys, there is a different key in the 128th bit, where the key is expressed in hexadecimal. The first set of random initial keys is shown in Table III. In the initial set of keys, there is a different key in the 128th bit, where the key is expressed in hexadecimal.

TABLE III INITIAL KEY

Key 1	b1 a0 43 57 d9 c5 66 f3 d4 e7 31 59 d8 a6 7f 2 <u>e</u>
Key 2	b1 a0 43 57 d9 c5 66 f3 d4 e7 31 59 d8 a6 7f 2 <u>f</u>

The results of the aggregation of the first group initial key after 10 rounds of key expansion are shown in Table IV.

TABLE IV BIT DIFFERENCE AFTER THE INITIAL KEY EXPANSION

Numb er of round s	Key 1	Key 2	Bit differe nce
initial	b1 a0 43 57 d9 c5 66 f3 d4 e7 31 59 d8 a6 7f 2 <u>e</u>	b1 a0 43 57 d9 c5 66 f3 d4 e7 31 59 d8 a6 7f 2 <u>f</u>	1
1	94 72 72 36 4d b7 14 c5 99 50 25 9c 41 f6 5a b2	94 72 56 36 4d b7 30 c5 99 50 01 9c 41 f6 7e b3	9
2	d4 cc 45 b5 99 7b 51 70 00 2b 74 ec 41 dd 2e 5e	d4 81 3b b5 99 36 0b 70 00 66 0a ec 41 90 74 5f	19
3	11 fd 1d 36 88 86 4c 46 88 ad 38 aa c9 70 16 f4	b0 13 f4 36 29 25 ff 46 29 43 f5 aa 68 d3 81 f5	52
4	48 ba a2 eb c0 3c ee ad 48 91 d6 07 81 e1 c0 f3	de 1f 12 73 f7 3a ed 35 de 79 18 9f b6 aa 99 6a	59
5	a0 00 af e7 60 3c 41 1a 28 ad 97 4d a9 4c 57 be	62 f1 10 3d 95 cb fd 08 4b b2 e5 97 fd 18 7c fd	71
6	a9 5b 01 34 c9 67 40 7e e1 ca d7 33 48 86 80 8d	ef e1 44 69 7a 2a b9 61 31 98 5c f6 cc 80 20 0b	59
7	ad 96 5c 66 64 f1 1c 18 85 3b cb 2b cd bd 4b a6	62 56 6f 22 18 7c d6 43 29 e4 8a b5 e5 64 aa be	63
8	57 25 78 db 33 d4 64 c3 b6 ef af e8 7b 52 e4 4e	a1 fa c1 fb b9 86 17 b8 90 62 9d 0d 75 06 37 b3	69
9	4c 4c 57 fa 7f 98 33 39 c9 77 9c d1 b2 25 78 9f	d5 60 ac 66 6c e6 bb de fc 84 26 d3 89 82 11 60	73
10	45 f0 8c cd 3a 68 bf f4 f3 1f 23 25 41 3a 5b ba	f0 e2 7c c1 9c 04 c7 1f 60 80 e1 cc e9 02 f0 ac	63

## V. EFFICIENCY ANALYSIS OF ENCRYPTION STANDARDS

Efficiency is a measure of the strengths and weaknesses of an algorithm. In addition to the limited performance of the perceived nodes in the IoT environment mentioned above, we value the operational efficiency of the designed encryption standard. Of course, the pursuit of high efficiency will inevitably affect the security of the algorithm, so this paper hopes to make the best balance between efficiency and security.

When testing the efficiency of the built encryption standard, this paper performs encryption and decryption tests on data of size 20KB, 40KB, 60KB, 80KB, and 100KB. The AES algorithm and the encryption standard DESI constructed in this paper can encrypt 128-bit data each time, and encrypt 20KB data as an example: 20KB is  $10 * 2^{11}$  B, that is,  $10 * 2^{14}$  bit. Each time 128-bit data is processed, 20KB of data needs to be cyclically encrypted.  $10 * 2^7$  times. For the same data, AES and DESI are used for encryption and decryption respectively. For each algorithm, the average value is obtained by multiple tests. The experimental results are shown in Table V.

TABLE V ADDITION AND DECRYPTION COMPARISON EXPERIMENT

Data size (KB)	AES runtime (ms)		DESI runtime (ms)	
	encrypt ion	Decryp t	encrypt ion	Decryp t
20	99	109	62	78
40	208	218.7	140	156
60	317.3	332.7	203	223.7
80	426.3	442	281	301.7
100	525.3	541	353.7	379.3

## VI. CONCLUSION

Based on several commonly used encryption algorithms in the Internet of Things, this paper optimizes the AES algorithm and combines the characteristics of IoT computing resources and storage resources to construct the data encryption standard DESI in the Internet of Things. The core work of this paper is to build the data encryption standard DESI in the Internet of Things based on the AES algorithm. The results show that DESI has higher efficiency than the AES algorithm. Combined with the security analysis of DESI, it can be concluded that DESI combines efficiency and security, and is suitable for providing encryption protection for data in the IoT environment.

## REFERENCES

- [1] [Chunling Sun. Application of RFID Technology for Logistics on Internet of Things[J]. AASRI Procedia,2012(1):106 – 111.
- [2] Almudena D'iaz-Zayas, Cesar A. Garc'ia-Pe'rez, A' lvaro M. Recio-Pe'rez. 3GPP standards to deliver LTE connectivity for IoT[C]. 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI),2016,283-288.
- [3] Wang Ying. Improvement of MixColumn() function in advanced encryption standard AES [D]. Shaanxi Normal University, 2011.
- [4] Christof Paar, Jan Pelzl. Exploring cryptography in depth - Principles and applications of commonly used encryption techniques [M]. Beijing: Tsinghua University Press, 2012.51-111.
- [5] Mary James, Deepa S Kumar. An Implementation of Modified Lightweight Advanced Encryption Standard in FPGA[J]. Procedia Technology, 2016(25):582 – 589.
- [6] Xiao Xiaocao, Li Shuguo. An Expression Method of S-Box and Inverse S-Box Replacement in AES Algorithm[J].Microelectronics & Computer,2014,31(1):112-115.
- [7] Mo Jianhua. Research on an encryption algorithm for WSN data security [D]. Zhejiang University of Technology, 2010.
- [8] Mojtaba Alizadeh, Wan Haslina Hassan, Mazdak Zamani, et al. Implementation and Evaluation of Lightweight Encryption Algorithms Suitable for RFID[J]. Journal of Next Generation Information Technology,2013,4(1):65-77