Joint AES and RSA for High-Definition Image Stream Safety with Key Managed

Gao Haifeng[1], Wang Kai[2,*] and Chen Shuizhong[1]

[1]*Department of Fire Control Electronics Research and Development, Luoyang Institute of Electro-optical Equipment, Luoyang, Henan, 471000, P.R. China*

[2]*School of Computer Science, Beijing Information Science and Technology University, Beijing, 100101, P.R. China*

Abstract: High definition images have been applied in more and more scenarios. Meanwhile, the demand for the safety of these image data is also increasing. In this paper, we firstly investigated the features existing in the high definition images and then studied the theory of the AES and RSA algorithms. However, the AES algorithm has some weaknesses when used, such as effectiveness in running and safety with used keys. In connection with the safety of image data and the used keys in the AES, we made some alterations on the AES algorithm and combined the AES and RSA algorithms to provide protection with centralized key management frame. For operating convenience, the paper introduces Data Stream Recognition mechanism to provide instruction on image stream with added stamps. After this, the safety of image data stream and the efficiency for handling images are both enhanced. Our outcome can be applied in data protection, especially when used in embedded operating system.

Keywords: High Definition Image, AES, Safety Protection, Managed Key, Data Stream Recognition, Algorithm, Stream.

1. INTRODUCTION

Rapid development of multimedia technology has prompted digital images being one of the most popular means used for network interaction. High-resolution images will become the mainstream in multimedia field in the future. However, image safety issues are paid great attention to by researchers. How to guarantee the security of digital image inter-action, and prevent image from being intercepted, tampered maliciously and illegal copying and trans-mission has been an important thing [1]. Image protection and transferring need a much more concern on the following questions.

1) Large amount of data. The information contained in the high definition will be much more enormous.

2) Strong real-time. In some specific application scenarios, the rapid rate of transmitting is required when some images are asked to convey.

3) Security transfer. In case of information security and preventing illegal users from access to image information, the need for high-resolution image encryption operations is required.

To achieve the safety protection, the high-definition images are required to be encrypted effectively. In modern cryptography, encryption algorithms can be divided into two categories: symmetric and asymmetric cryptography. Because of the high rate in systemic cryptography, it is more suitable for encryption of large scale data.

AES algorithm is an efficient approach to provide safety freely and easily. It can be applied to encrypt a large amount of image data. However, the key used in AES should be managed effectively. Otherwise, once the key is accessed maliciously by a third party, the image data which has been encrypted is meaningless. In general, the used keys can be protected by some cryptography algorithms. RSA is one of these algorithms, being a kind of public key encryption algorithm [2]. The RSA can be used to encrypt short message, being safe and easy to understand and implement. In current applications, it is the most popular cryptography algorithm.

In this paper, we select the AES to provide safety for the large amount of image data. As well, the RSA is applied to provide security for the keys used in AES algorithm. Some transform has been made about AES to satisfy the use in large scale data scenario. Except for it, centralized key management and Data Stream Recognition mechanism are introduced to the case of efficient key management.

2. AES ALGORITHM

There are some symmetric cryptographic algorithms which are popular, such as DES and AES. However, DES has already not satisfied the existing security requirements along with the development of the society and science and technology.

In this case, NIST published the Advanced Encryption Standard. The input group and output group in the AES algorithm both have a length of 128 bits. And it is described by a matrix in a shape of square and in the units of bytes. In the whole algorithm, the key which is imported is expanded into an array. This array is composed by 44 units and each unit

*Address correspondence to this author at the School of Computer Science, Beijing Information Science and Technology University, Beijing, 100101, P.R. China; E-mail: clomis@foxmail.com

has a word of 32 bits. In every stage of both encryption and decryption process, there is a key with four words used for round key.

The input group transforms the data into a four mix four matrix *State*, which will be changed in every stage of encryption and decryption process. Also, the key with a length of 128 bits is described by a column matrix by unit of bytes. Since then, this key will be expanded into a key sequence array which has a unit of words. And every word is composed by four bytes. Finally, the input key with a length of 128 bits has been expanded into a sequence with a length of 44 words. The sequence matrix ***w*** will be used in every round of both encryption and decryption process.

There are four main basic steps: SubBytes, ShiftRows, MixColumns, and AddRoundKey [3]. The SubBytes is a non-linear byte substitution which operates independently on each byte of *State* by S-box table [4]. S-box table contains 256 numbers (from 0 to 255), composed by a 16*16-byte matrix.

In SubBytes, every byte in *State* will be mapped to a new byte by the following approach. The high four bits in the byte will be extracted as the value of row, while the low four ones as the value of column. The element in S-box will be selected by the two values and changes into the output.

The RowShifts is an operation of shifting based on *State*. The first line in the matrix remains unchanged. The second line will be rotated left by one byte, and so forth. The operation will be much more useful than it looks [5]. It removes one specific byte of the column into another column by a linear distance, which ensures that the four bytes in a column are expanded into four different columns respectively.

MixColums operates on every column independently, as the Equation (1). Here the previous matrix is expressed as ***s*** while the result as ***s'***. Every element in the product matrix is the sum of these products of the row data and the corresponding column data. Here, the multiplication and addition are both defined on *GF (2^8)* [6]. The MixColumns of column *j* in *State* can be expressed as Equation (1).

The AddRoundKey does an operation of XOR between the four bytes of one specific column in *State* and one word of the round key. It has an impact on every bit in the *State* matrix in spite of the simple. The complexity of key expanding and other stages in the AES algorithm both ensure the security of the algorithm.

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} = \begin{pmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{11} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{pmatrix} \quad (1)$$

3. RSA ALGORITHM

For the image data is encrypted by systemic keys, once the AES key is stolen, it means that the privacy data will be exposed to the third party. So how to provide the security for the image data has been a question. The solution will be asymmetric algorithm. For its encryption key is different from its decryption key. RSA is probably the most recognizable asymmetric algorithm [7].

Theory foundation about RSA is the factor decomposition difficulty of a big sum Numbers in number theory. It means that striving for the product of two large prime Numbers on a computer is easy to implement, while solving a large sum Numbers into multiplication of two prime Numbers is difficult. RSA encryption algorithm holds two keys being related but different. One is called public key, the other, private key. In public key cryptographic algorithms, encryption keys can be pronounced as a public key, and the decryption key have to be hidden as a private key. In the encryption algorithm, both public and secret keys can be used for data encryption. The one is used for encryption, and the other one can be used as the corresponding decryption key. The private key cannot be deduced from the public key, and plain text cannot be deduced from the cipher text either.

RSA cryptosystem uses the module n, the smallest non-negative completes the remaining lines of operation, where n is the product of two different primes *P* and *Q* [8]. RSA algorithm is described as follows.

Firstly, the procedure of keys generation is as follows.

1) Randomly generate two primes *P* and *Q* of length *K/2* bits;

2) Calculate the public key *Key= P*Q* and the length of public key is k-bit;

3) Generate a random encryption key named *E*, subject to $2<= E<= \Phi(n)-1$ and $GCD(E,\Phi(n))= 1$. This is the necessary and sufficient conditions for solvability of the decryption key. Assume that the decryption key is expressed as *D*. Then $E*D \bmod \Phi(n)= 1$. $\Phi(n)$ is known as the Euler function of n, and the value is $\Phi(n) = (P-1)*(Q-1)$.

4) Calculate the decryption key, $D = (E-1) \bmod n$, $E-1$ is inverse for the decryption key *D*. The formula of the original equation is $(E*D) \bmod \Phi(n)=1$.

Now, the encryption key and decryption key have been created. The encryption process of the plain text and decryption process of cipher text can be expressed as follows. Assume the plain text is *M* and the cipher text is *C*.

1) Encryption: *C = M^E mod Key*;

2) Decryption: *M = C^D mod Key* [9].

4. APPLICATION AND SIMULATION

When applied in specific project, AES algorithm has also some weaknesses [9]. In the process of encryption and decryption, there are a large amount of calculations costing much time. To simplify the two processes and enhance the efficiency has been the key question. Meanwhile, the keys used in the AES process calls for protection. This can be called the issue of key management. Finally, when running in real development board, the application must take a cost of buffer to provide an efficient performance. To select an appropriate buffer-size is also an important point.

4.1. Optimization on AES

As for the first issue, we optimize the implement process of the AES. Assume that the matrix *State* is denoted by $a_{i,j}$ and the round key matrix is represented by $k_{i,j}$. The conversion can be described as follows. Now we will derive the transformation. We have assumed that $a_{i,j}$ represents the matrix State before handled. Similarly, $b_{i,j}$ represents the matrix *State* after the stage SubBytes and $c_{i,j}$ after ShiftRows and $d_{i,j}$ after MixColumns and $e_{i,j}$ after AddRoundKey. We can see that the $e_{i,j}$ is just the output in this round. The basic transformation is shown in Table **1**.

$$\begin{pmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{pmatrix} = \begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} \oplus \begin{pmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{pmatrix} =$$

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \bullet \begin{pmatrix} s[a_{0,i}] \\ s[a_{1,j+1}] \\ s[a_{2,j+2}] \\ s[a_{3,j+3}] \end{pmatrix} \oplus \begin{pmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{pmatrix} \tag{2}$$

$$= \begin{pmatrix} 02 \\ 01 \\ 01 \\ 03 \end{pmatrix} \bullet s[a_{0,j}] \oplus \begin{pmatrix} 03 \\ 02 \\ 01 \\ 01 \end{pmatrix} \bullet s[a_{1,j+1}] \oplus \begin{pmatrix} 01 \\ 03 \\ 02 \\ 01 \end{pmatrix} \bullet s[a_{2,j+2}]$$

$$\oplus \begin{pmatrix} 01 \\ 01 \\ 03 \\ 02 \end{pmatrix} \bullet s[a_{3,j+3}] \oplus \begin{pmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{pmatrix}$$

$$Te0[x] = \begin{pmatrix} 02 \\ 01 \\ 01 \\ 03 \end{pmatrix} \bullet s[x] \quad Te1[x] = \begin{pmatrix} 03 \\ 02 \\ 01 \\ 01 \end{pmatrix} \bullet s[x]$$
$$Te2[x] = \begin{pmatrix} 01 \\ 03 \\ 02 \\ 01 \end{pmatrix} \bullet s[x] \quad Te3[x] = \begin{pmatrix} 01 \\ 01 \\ 03 \\ 02 \end{pmatrix} \bullet s[x] \tag{3}$$

Now we can formulate one standard round as Equation (2).

From the above derivation, we can conclude that one round can be expressed as Equation (3). Now we can define four tables which have a length of 256 words.

Every time we get a specific byte of data, we can search the four tables to get the results. Absolutely, looking up on tables is much faster than calculating.

Table 1. Basic transformation

Step	Transformation
SubBytes	$b_{i,j} = s[a_{i,j}]$
ShiftRows	$\begin{pmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{pmatrix} = \begin{pmatrix} b_{0,j} \\ b_{1,j+1} \\ b_{2,j+2} \\ b_{3,j+3} \end{pmatrix}$
MixColumns	$\begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \bullet \begin{pmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{pmatrix}$
AddRoundKey	$\begin{pmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{pmatrix} = \begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} \oplus \begin{pmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{pmatrix}$

4.2. Protection for AES Keys

The application introduced RSA to solve the issue of key management. There are two types of keys, named AES keys and RSA keys. While AES keys are used to encrypt and decrypt image data, RSA keys provide safety for AES keys.

AES algorithm has an advance on rapid encrypting speed and high encryption intensity. However, there will be potential safety hazard when single key is used. Meanwhile, safe key distribution mechanism in RSA algorithm is appealing to solve the issue even though the speed is relatively low [10].

The application is composed by server part and client part. It adapts centralized key management where server part is responsible for key-generation and key-protection and client part focuses on key-restoration and key-using. Every key owns a unique 16-byte stamp. After being generated, AES keys will be encrypted by the selected RSA key. The results will be composed by stamp of the AES key and RSA key, cipher AES key and other relative information. In this way, the cipher key will instruct which plain AES key it is and which RSA key was used to encrypt it. Once connecting to the server, the specific client will obtain a unique stamped RSA public key. And then, server will send cipher AES key periodically if necessary and the specific client could be able to acquire the plain AES-key by decryption with its RSA key.

4.3. Protection for AES Keys

After the client connected with server, the received AES key and other subsequent AES keys in client are all protected by the unique RSA-key.

For the coming image stream data, client encrypts it with current AES key. The stamp plays a critical role in the whole process, named Data Stream Recognition mechanism (DSR). Once the used AES key changed, the application will add

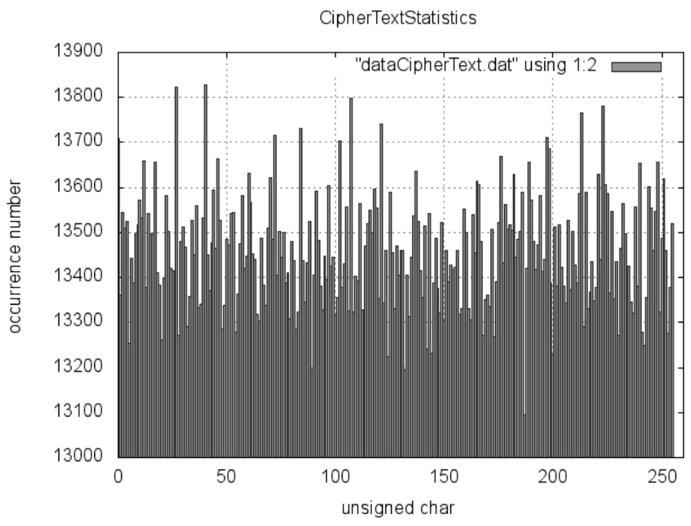

Fig. (1). Statistics information before encryption.

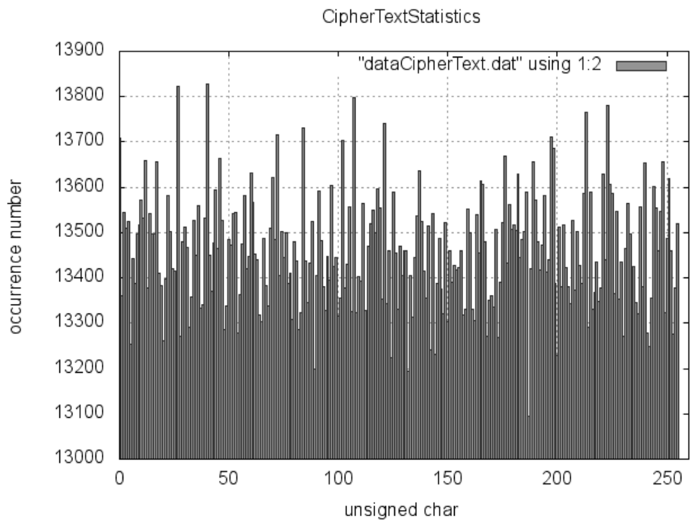

Fig. (2). Statistics information after encryption.

new key's stamp before the output stream in encryption process. Before the stamp, there are two stuffing bytes added to the output to instruct one stamp is following.

As for decryption process, once encountering a stamp, the application will handle the forward data firstly and then update AES key by the stamp. This handle in decryption is more complex than that in encryption.

In addition, the DSR mechanism requires a specific length of buffer size, constrained by AES algorithm where in-put group must be 128-bit length [11]. As a sequence, there must be a pre-handled process to make data length satisfy the character. Note that the appropriate buffer size will be discussed in the next section. Before encrypted, the image data stream will meet some statistical law which could be used by third party to access the confidential information improperly. This statistics in image data after encryption will be mitigated. The difference can be compared with Figs. (**1** and **2**). With these process and optimization, the safety and the efficiency both can be achieved.

4.4. Selection of Buffer Size

The application will require large scale buffer size when operating. Different size makes different performance. If the size is small, the handling speed of stream data will be much lower than the speed of application running, thus more and more data could accumulate. In contrast, application performance may be efficient enough, but it could degrade the whole system performance. So to select an appropriate buffer size becomes a critical issue. Following some simulations with different buffer size, we can choose a suitable size to apply according to the handling rate of application. In application, 1MB has been chosen as the appropriate buffer size.

CONCLUSION

In this paper, we have studied the algorithm of AES and RSA. And we take the AES into the real usage by protection of RSA. We make some optimization about the AES algorithm, which makes it suitable to be applied in high-definition image data scenario. To solve the issue of key management and data stream recognition, we introduce centralized key management scheme and DSR mechanism which provide high safety intensity. The hardware implementation of the Rijndael algorithm can provide either high performance or low cost for specific applications [12], which is the next step in the series of work.

CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] X. Zhang, M. Wang, and G. Zhu, "Research on the new development of image encryption algorithms," *Computer Engineering & Science*, vol. 34, pp. 1-6, 2012.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120-126, 1978.

[3] T. Hoang, "An efficient FPGA implementation of the Advanced Encryption Standard algorithm," In: *Proceedings of IEEE RIVF International Conference on Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF)*, Ho Chi Minh City, 2012, pp. 1-4.

[4] H. S. Deshpande, K. J. Karande, and A. O. Mulani, "Efficient implementation of AES algorithm on FPGA," In: *Proceedings of International Conference on Communications and Signal Processing (ICCSP)*, Melmaruvathur, 2014, pp.1895-1899.

[5] M. Tunstall, D. Mukhopadhyay, and S. Ali, "Differential fault analysis of the advanced encryption standard using a single fault," In: *Proceedings of the 5th IFIP WG 11.2 International Conference on Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication*, vol. 6633, pp. 224-233, 2011.

[6] S. William, and W. Stallings, *Cryptography and Network Security*, 4/E. Pearson Education India, 2006.

[7] U. Somani, K. Lakhani, and M. Mundra, "Implementing digital signature with rsa encryption algorithm to enhance the data security of cloud in cloud computing," In: *Proceedings of 1st International Conference on Parallel Distributed and Grid Computing (PDGC)*, Solan, 2010, pp. 211-216.

[8] J. H. Hong, and C. W. Wu, *RSA Public Key Crypto-Processor Core Design and Hierarchical System Test Using IEEE 1149 Family*, PhD Thesis, National Tsing-Hua University, Taiwan, 2000.

[9] S. H. Kamali, R. Shakerian, M. Hedayati, and M. Rahmani, "A new modified version of advanced encryption standard based algorithm for image encryption," In: *Proceedings of International Conference On Electronics and Information Engineering (ICEIE)*, Kyoto, 2010, pp. V1-141 - V1-145.

[10] J. Dong, F. Song, and W. Guan, "A hard disk partition encryption techniques base on aes&rsa mixed encryption strategy," *Microelectronics & Computer*, vol. 29, pp. 135-137, 2012.

[11] J. Han, J. Lin, and W. Zhou, "Design of AES core based on FPGA," *Computer Engineering & Science*, vol. 35, pp. 80-84, 2013.

[12] J. Balamurugan, and E. Logashanmugam, "Low power and high speed aes using mix column transformation," In: *Proceedings of International Conference on Current Trends in Engineering and Technology*, Coimbatore, 2013, pp. 216-219.

Received: August 16, 2015 Revised: September 12, 2015 Accepted: September 22, 2015