Enhanced Hybrid Security Protocol Based On AES and RSA

Kai-Yu Lu
Xi'an Jiaotong-liverpool University (XJTLU)
Suzhou, Jiangsu, China
KaiYu.Lu16@student.xjtlu.edu.cn
ID: 1614649

Abstract—**In the information era, most of communications are implemented by the advanced technologies, which makes the security become a central problem. This paper comprehensively analyzes the characteristics of AES and RSA algorithms, such as security level, encryption speed and key distribution, and proposes a hybrid security protocol based on AES and RSA by utilizing respective superiorities to enhance shortcomings with each other. The proposed security protocol is implemented by using Python programming language on the Microsoft Visual Studio 2019. The communication was implemented by TCP connection. Finally, the testing result show that the proposed protocol is implemented successfully and has high speed which is 0.026413 seconds in average.**

Keywords—AES, RSA, Secret Key, Public Key, Private Key, encryption, decryption.

I. Introduction

Recently, the Internet has become the most significant communication tools and the security of the data transmission via the Internet has attracted considerable attention. Hence, numerous security protocols have been proposed to protect the data privacy from unauthorized third parties. There are plentiful methods to guarantee the security, different types of data such as image, audio and e-mails can be protected by encryption which is one of the effective security solutions. Encryption can be assumed as a process of converting readable and understandable information into incomprehensible data. On the contrary, decryption is to convert the incomprehensible data into original information. There are two encryption techniques in data encryption, which are asymmetric encryption and symmetric encryption. Symmetric encryption utilizes one secret key to encrypt in the transmitter and decrypt in the receiver. Since the secret key is shared between both sides, confidentiality is prerequisite to be retained. Advanced Encryption Standard (AES) is commonly used as a symmetric algorithm and has superior usability and robustness [1]. With respect to the asymmetric encryption, a public and a private key are utilized for encrypting and decrypting. The public key can be shared to every side and is utilized to encrypt the plaintext while the private key retains secret and is used for decrypting the encrypted plaintext. Rivest-Shamir-Adleman (RSA) is one of the famous techniques in the asymmetric encryptions [1].

According to [1], although the AES algorithm has been proved to be one of the most powerful and effective algorithms, the distribution of the secret keys becomes a pivotal problem [1]. Since AES utilizes the fixed key to encrypt and decrypt the information, once the secret key in one side was changed, another side should be informed to be change the key as specified, which becomes infeasible to use. Therefore, AES is quite suitable for encrypting long information. With respect to RSA algorithm, it is assumed to have higher security but require longer encryption time and memory usage [2] [3]. Therefore, RSA is suitable for encrypting short information and provide high security.

This project intends to propose a hybrid security protocol based on AES and RSA by utilizing respective superiorities to enhance shortcomings with each other. The rest of the paper is organized as follow. Section II describes the working principles of AES and RSA. Problems in AES and RSA will be identified in section III. The proposed solution is elaborated in section IV. Section V will describe the detailed solutions with implementations and show the testing results. Finally, section VI will present a conclusion for this paper.

II. Related Work

Since Advanced Encryption Standard (AES) has characteristics of easy, fast to implement, high robustness, more challenging to be attacked and so on, it has been widely applied in the area of security communication. AES is a symmetric algorithm, also called Rijindael, which is based on four transformations iterations consisting of SubByte, ShiftRow, MixColumn and AddRoundKey [4]. According to Rijindael, the number of iterations also called encryption or decryption rounds depends on selected types of the key length. There are three types of the key length in the AES algorithm: 128, 192, and 256 bits, which are used for encrypting and decrypting a fixed 128-bit grouped data blocks [4]. The specific relationship between the key length and AES algorithm iterations is summarized in Table 1.

AES Variant	**AES algorithm structure**		
	Key length	**Number of iterations**	**Number of 32-bit words**
AES-128	128	10	4
AES-192	192	12	6
AES-256	256	14	8

Table 1: Relationship between key length and AES algorithm iterations [4] [5]

As previously mentioned, there are four transformations involved in each round of AES encrypted loop:

- **SubByte:** SubByte transformation is the first operation that provides a non-linear function for each state array cell to be substituted with the corresponding bytes. The corresponding bytes are pre-defined in a matrix called substitution box whose size is 15 x 15 [4] [6].

- **ShiftRow:** ShiftRow circularly shifts each row to arrange the state in matrix, which is a permutation operation [6].
- **MixColumn:** MixColumn is a substitution stage which uses a fixed matrix of 4 x 4 to fully mixes each row in a matrix by a linear transformation [4] [6].
- **AddRoundKey:** Every byte in the matrix is operated XOR with a round key at each loop.

Finally, the output through AES encryption would become the encrypted data in a final state array [7]. The flowchart of AES-128 algorithm for 10 rounds is summarized in Figure 1.

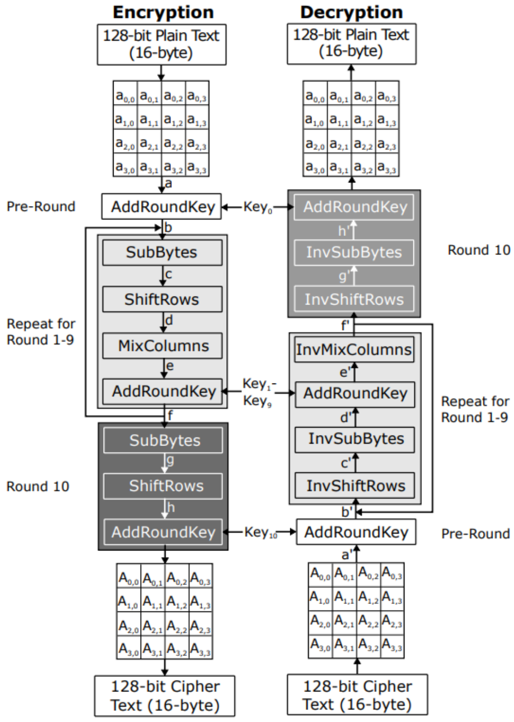

Encryption
Decryption
128-bit Plain Text (16-byte)
128-bit Plain Text (16-byte)
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
Pre-Round
AddRoundKey
Key_0
a
b
SubBytes
c
ShiftRows
d
MixColumns
e
AddRoundKey
Key_1-Key_9
Repeat for Round 1-9
f
Round 10
SubBytes
g
ShiftRows
h
AddRoundKey
Key_{10}
$A_{0,0}$ $A_{0,1}$ $A_{0,2}$ $A_{0,3}$
$A_{1,0}$ $A_{1,1}$ $A_{1,2}$ $A_{1,3}$
$A_{2,0}$ $A_{2,1}$ $A_{2,2}$ $A_{2,3}$
$A_{3,0}$ $A_{3,1}$ $A_{3,2}$ $A_{3,3}$
128-bit Cipher Text (16-byte)
AddRoundKey
h'
InvSubBytes
g'
InvShiftRows
Round 10
f'
InvMixColumns
e'
AddRoundKey
d'
InvSubBytes
c'
InvShiftRows
b'
Repeat for Round 1-9
AddRoundKey
Pre-Round
a'
128-bit Cipher Text (16-byte)

Figure 1: 10-round AES-128 encryption/decryption flowchart

AES has certain significant crypto modes including: cipher-block chaining (CBC), electronic code book (ECB), counter (CTR), cipher feedback (CFB) and output feedback (OFB). It is stated that the mode of CBC provides substantial security has higher security level than ECB [9]. In EBC mode, the data is split in to multiple blocks with separately encrypted. Therefore, the identical ciphertext blocks are encrypted form the identical plaintext, which forms a drawback generating vulnerabilities described in [9]. In order to solve this problem, CBC combine the data in previous ciphered blocks with an initialization vector (IV). Additionally, the block is operated XOR with the previous ciphertext block. Consequently, the encrypted results are always unique. The encryption and decryption of the CBC mode is provided in Figure 2 and Figure 3 respectively.

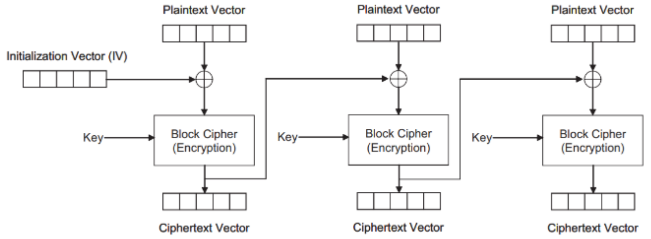


Figure 2: Encryption of CBC mode

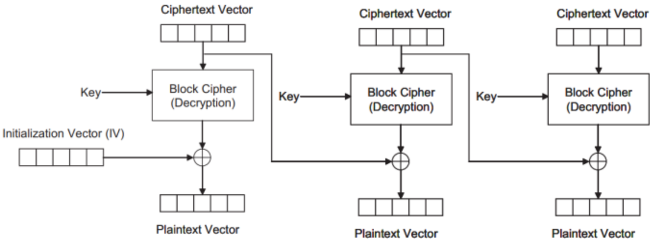


Figure 3: Decryption of CBC mode

RSA is a public-key cryptography (asymmetric cryptography) proposed by Ron Rivest, Adi Shamir and Leonard Adelman in 1978. RSA is assumed as a secure technique used for protecting data from attacks by encryption, exchanging keys and digital signature [7]. In RSA, the public key is used for encrypting the plaintext and can be shared to any user via a secure channel. However, the public key cannot decrypt the data except the private key with the authenticated receiver. There are three pivotal steps in RSA algorithm, which are key generation, encryption and decryption. The flowchart for explaining the working principle are indicated in Figure 4.

- **Key Generation:** There are two keys are generated in this step, which are public key and private key. The public key is shared in public for encryption while the private key is available for the receiver only for decryption. The process for generating these two keys works as below.

a) Randomly select two distinct prime numbers p and q.

b) Compute $n = p \times q$, where n is the modulus for the public key and the private key.

c) Compute $\phi(n) = (p-1)(q-1)$.

d) Select an integer e such that $GCD\big(e, \phi(n)\big) = 1$, where $1 < e < \phi(n)$.

e) Solve the equation $e * d = 1$ and obtain the variant d.

f) (n, e) pair is the public key and (d, n) is the private key.

- **Encryption:** $c = m^e \ mod \ n$, where c is the cipher text and m is the plaintext.
- **Decryption:** $m = c^d \ mod \ n$

III. PROBLEM IDENTIFICATION

As previously mentioned, AES has been proved as a fast and secure encryption algorithm. However, AES has problems in key distribution and scalability. Therefore, AES is suitable for encrypting long information. With respect to RSA, it has higher security level than AES. Nonetheless, longer encryption time and memory usage are required when applying RSA algorithm. Therefore, RSA is suitable for encrypting short information. Based on this, a complementary solution by combining AES and RSA is proposed in this paper, which will be explained in Section IV.

IV. Proposed Solution And Novelty

Since RSA generated a public key and a private key, it performs better key distribution than AES. However, AES has faster encryption performance than RSA. Therefore, this project intends to hybridize AES and RSA to produce a security protocol having higher security and faster encryption. In this security protocol, RSA is responsible for generating a pair of keys (public key and private key) and transmitting the public key to the sender. In the sender, AES generates a secret key to encrypt the plaintext. Meanwhile, the AES secret key will be encrypted by the public key from the receiver. Consequently, both encrypted plaintext and encrypted AES secret key are transmitted to the sender. After this, encrypted AES secret key is decrypted by the original RSA private key and then the AES secret key is obtained. Finally, the encrypted plaintext will be decrypted by the obtained AES secret key. The flowchart for the proposed security protocol is illustrated in Figure 5.

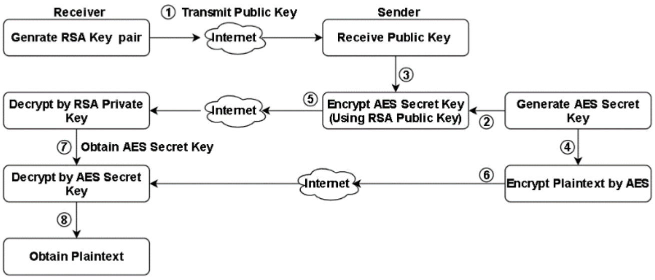
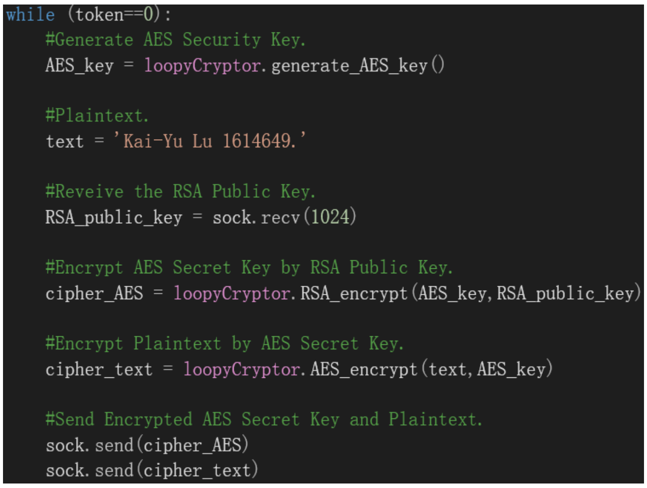


Figure 5: Flowchart of the proposed security protocol.

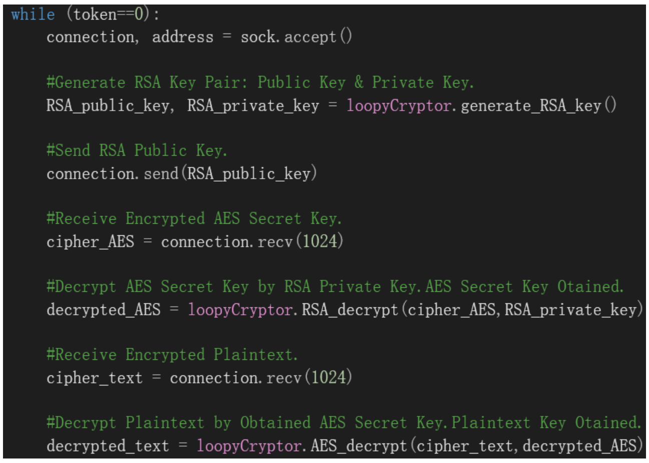
V. Implementation and Testing

In this project, the hybrid security protocol based on AES and RSA was verified and implemented by using Python programming language on the Microsoft Visual Studio 2019. The communication was implemented by TCP connection. There several parts involved in the implementation: RSA key pair generation, AES secret key encrypted by RSA public key, plaintext encrypted by AES secret key, encrypted AES secret key decrypted by RSA private key, encrypted plaintext decrypted by AES secret key. The core parts of implementation for the sender and the receiver are provided in Figure 6 and Figure 7 respectively.



```
while (token==0):
    #Generate AES Security Key.
    AES_key = loopyCryptor.generate_AES_key()

    #Plaintext.
    text = 'Kai-Yu Lu 1614649.'

    #Reveive the RSA Public Key.
    RSA_public_key = sock.recv(1024)

    #Encrypt AES Secret Key by RSA Public Key.
    cipher_AES = loopyCryptor.RSA_encrypt(AES_key,RSA_public_key)

    #Encrypt Plaintext by AES Secret Key.
    cipher_text = loopyCryptor.AES_encrypt(text,AES_key)

    #Send Encrypted AES Secret Key and Plaintext.
    sock.send(cipher_AES)
    sock.send(cipher_text)
```

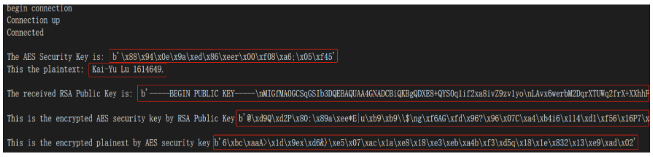
Figure 6: Implementation for the sender



```
while (token==0):
    connection, address = sock.accept()

    #Generate RSA Key Pair: Public Key & Private Key.
    RSA_public_key, RSA_private_key = loopyCryptor.generate_RSA_key()

    #Send RSA Public Key.
    connection.send(RSA_public_key)

    #Receive Encrypted AES Secret Key.
    cipher_AES = connection.recv(1024)

    #Decrypt AES Secret Key by RSA Private Key. AES Secret Key Otained.
    decrypted_AES = loopyCryptor.RSA_decrypt(cipher_AES,RSA_private_key)

    #Receive Encrypted Plaintext.
    cipher_text = connection.recv(1024)

    #Decrypt Plaintext by Obtained AES Secret Key. Plaintext Key Otained.
    decrypted_text = loopyCryptor.AES_decrypt(cipher_text,decrypted_AES)
```

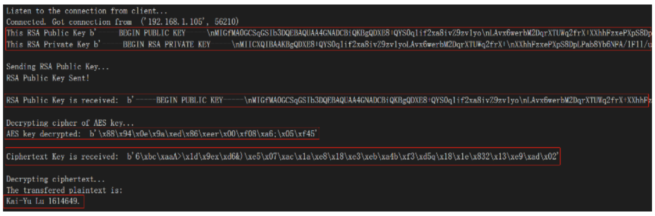
Figure 7: Implementation for the receiver

The used plaintext in this project is “Kai-Yu Lu 1614649”. In order to verify the correctness of this protocol, the experimental results in the sender and the receiver are shown in Figure 8 and Figure 9 respectively.



```
begin connection
Connection up
Connected

The AES Security Key is:  b'\x88\x94\x0e\x9a\xed\x86\xeer\x00\xf08\xa6;\x05\xf45'
This the plaintext:  Kai-Yu Lu 1614649.

The received RSA Public Key is:  b'-----BEGIN PUBLIC KEY-----\nMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDXE8+QYS0q1if2xa8ivZ9zv1yo\nLAvx6werbM2DqrXTUWq2frX+XXhhF

This is the encrypted AES security key by RSA Public Key b'@\xd9Q\xd2P\x80:\x89a\xee*E|u\xb9\xb9\\$\ng\xf6AG\xfd\x96?\x96\x07C\xa4\xb4i6\x114\xd1\xf56\x16P7\x

This is the encrypted plainext by AES security key b'6\xbc\xaaA>\x1d\x9ex\xd6&)\xe5\x07\xac\x1a\xe8\x18\xe3\xeb\xa4b\xf3\xd5q\x18\x1e\x832\x13\xe9\xad\x02'
```

Figure 8: Testing result in the sender



```
Listen to the connection from client...
Connected. Got connection from  ('192.168.1.105', 56210)
This RSA Public Key b'-----BEGIN PUBLIC KEY-----\nMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDXE8+QYS0q1if2xa8ivZ9zv1yo\nLAvx6werbM2DqrXTUWq2frX+XXhhFzxePXpS8Dp
This RSA Private Key b'-----BEGIN RSA PRIVATE KEY-----\nMIICXQIBAAKBgQDXE8+QYS0q1if2xa8ivZ9zv1yoLAvx6werbM2DqrXTUWq2frX+\nXXhhFzxePXpS8DpLPab8Yb6NFA/1F11/u

Sending RSA Public Key...
RSA Public Key Sent!

RSA Public Key is received:  b'-----BEGIN PUBLIC KEY-----\nMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDXE8+QYS0q1if2xa8ivZ9zv1yo\nLAvx6werbM2DqrXTUWq2frX+XXhhFz

Decrypting cipher of AES key...
AES key decrypted:  b'\x88\x94\x0e\x9a\xed\x86\xeer\x00\xf08\xa6;\x05\xf45'

Ciphertext Key is received:  b'6\xbc\xaaA>\x1d\x9ex\xd6&)\xe5\x07\xac\x1a\xe8\x18\xe3\xeb\xa4b\xf3\xd5q\x18\x1e\x832\x13\xe9\xad\x02'

Decrypting ciphertext...
The transfered plaintext is:
Kai-Yu Lu 1614649.
```

Figure 9: Testing result in the receiver

From Figure 8 and Figure 9, it can be seen that the AES security key before encryption and after decryption are identical. Additionally, RSA public key, encrypted plaintext and AES security key are transferred successfully. The encrypted AES security key encrypted by the RSA public key can be decrypted by the RSA private key according to Figure 8 and Figure 9. Finally, the plaintext “Kai-Yu Lu 1614649” is transferred successfully.

In order to verify the speed of the proposed protocol, several tests have been conducted. The testing results of the speed performance is summarized in Table 2.

Test	Protocol Running Time (s)
1	0.027925
2	0.023907
3	0.025896
4	0.027925
Average	0.026413

Table 2: Test results of the protocol running time

It can be observed that the protocol can be finished in a quite short time, which proves that the respective efficiency of AES and RSA can be enhanced by combing them together.

VI. Conclusion

This paper has thoroughly introduced the encryption techniques of AES and RSA and indicated their respective superiorities and drawbacks by sufficient literature researches. A hybrid security protocol based on AES and RSA is proposed and implemented successfully in this project . Although there exists possibility that the RSA public key and ciphertext could be obtained illegally by the unauthorized third-party, the real content cannot be viewed or modified. In conclusion, this security protocol utilizes RSA technique to transmit the AES security key and encrypts the plaintext by the AES security, which not only solves the problem of AES key distribution, but also achieves a highly efficient encryption performance. This project can be an inspiration that the drawbacks from every single encryption technique might be solved by compensated with each other.

References

[1] D. M. Alsaffar *et al.*, "Image Encryption Based on AES and RSA Algorithms," in *Int. Conf. on Computer Applications & Information Security* (ICCAIS 2020), Riyadh, Saudi Arabia, 2020, pp. 1-5.

[2] A. Chandel *et al.*, "Comparative Analysis of AES & RSA Cryptographic Techniques," in *Int. Conf. on Computational Intelligence and Knowledge Economy* (ICCIKE 2019), Dubai, United Arab Emirates, 2019, pp. 410-414.

[3] E.Thambiraja *et al.*, "A survey on various most common encryption techniques." In *Int. J. of advanced research in computer science and software engineering*, vol. 2, no.7, pp 226-233, Feb. 2012.

[4] Y. Zhu *et al.*, "Study of the AES Realization Method on the Reconfigurable Hardware," in *2013 Int. Conf. on Computer Sciences and Applications*, Wuhan, 2013, pp. 72-76.

[5] N. Floissac and Y. L'Hyver, "From AES-128 to AES-192 and AES-256, How to Adapt Differential Fault Analysis Attacks on Key Expansion," in *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, Nara, 2011, pp. 43-53.

[6] Ritambhara *et al.*, "An enhanced AES algorithm using cascading method on 400 bits key size used in enhancing the safety of next generation internet of things (IOT)," in *Int. Conf. on Computing, Communication and Automation* (ICCCA 2017), Greater Noida, 2017, pp. 422-427.

[7] B. J. S. Kumar et al., "Comparative study on AES and RSA algorithm for medical images," in *Int. Conf. on* Communication and Signal Processing (ICCSP 2017), Chennai, 2017, pp. 0501-0504.

[8] L. Yu et al., "AES Design Improvements Towards Information Security Considering Scan Attack," in *IEEE Int. Conf. on Trust, Security And Privacy In Computing And Communications/IEEE International Conference On Big Data Science And Engineering* (TrustCom/BigDataSE 2018), New York, NY, 2018.

[9] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C.* UK: Wiley, 2015.