EAI Endorsed Transactions on Security and Safety

Research Article **EAI.EU**

A Multi-connection Encryption Algorithm Applied in Secure Channel Service System

Fanhao Meng[1,2], Rongheng Lin[1,2,*], Zhuoran Wang[1,2], Hua Zou[1], Shiqi Zhou[1,2]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications
[2] Science and Technology on Information Transmission and Dissemination Networks Laboratory

Abstract

Encryption is the most important method to enhance security of network transmitting. SDN (Software Defined Networking) Security Transmission Service can provide multi-connection transmitting service, which scatters data to multiple network connections for transmission so that data on different connections is isolated from each other. Based on the service, encrypting the isolated data prevents overall data from intercepted and deciphered. In the above scenario, we propose an encryption algorithm that uses the data themselves as encryption keys, and use the data isolation effect of multi-connection transmission to distribute the encrypted ciphertext to different network transmission paths, which is equivalent to using a rather random sequence as an encryption key for each data fragment without sharp increase in transmitting data, so that data transmitted on every connection are ensured to be safe. After compared with other encryption algorithms such as DES, AES and RSA, it is proved that in the multi-connection transmitting scenario this algorithm has better encryption effect and operating efficiency, which provides an effective guarantee for network security.

Keywords: encryption, multi-connection transmission, sequence cipher.

Received on 28 April 2018, accepted on 13 May 2018, published on 09 October 2018

doi: 10.4108/eai.15-5-2018.155167

1. Introduction

To ensure data security in the network transmission, we usually use data encryption to prevent information disclosure. Cryptography is divided into symmetric encryption algorithms and asymmetric algorithms, and symmetric algorithms have two categories: block cipher and sequence cipher. DES, AES and RSA are commonly used encryption algorithms [1]. Aiming at risks of data transmitting on networks, this paper proposes a data encryption strategy collaborating with multi-connection transmission. Multi-connection transmission is to slice data into multiple data flows to transmit on multiple TCP connections. The sender sends data through a number of ports according to a defined strategy, and the receiver will receive packets from the same number of ports and recombine data fragments into the original data according to the slicing strategy. Multi-connection transmission is one of the security strategies of SDN Security Transmission Service, which uses distribution on network transport layer to separate data, cooperating with time-slot transmission strategy, which is another strategy of SDN Security Transmission Service to ensure data security by switching switch paths of forwarding packets in every different time slot. It can change the order of transmitted data and scatter packets to multiple physical paths. So even if some of network equipment has been hacked or data transmitted on some links has been intercepted, the possibility of information leakage will be the least.

Based on the multi-connection transmission, the encryption strategy proposed in this paper has the characteristics of simple and efficient, and can guarantee the security of each data fragment without increasing the amount of transmission data. In the scenario of multi-connection combined with time-slot transmission, using this encryption algorithm can almost absolutely guarantee the security of the

*Corresponding author. Email: rhlin@bupt.edu.cn

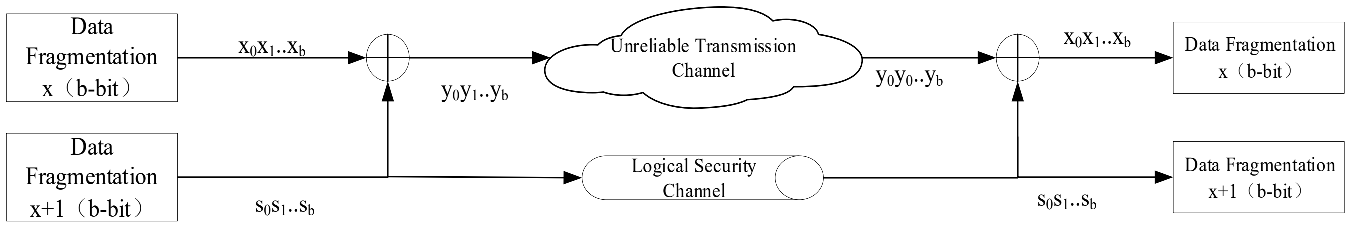
Data Fragmentation x（b-bit）
$x_0x_1..x_b$
$y_0y_1..y_b$
Unreliable Transmission Channel
$y_0y_0..y_b$
$x_0x_1..x_b$
Data Fragmentation x（b-bit）
Data Fragmentation x+1（b-bit）
$s_0s_1..s_b$
Logical Security Channel
$s_0s_1..s_b$
Data Fragmentation x+1（b-bit）

Figure 1. Multi-connection Encryption Transmission Diagram

data transmitted on the network. This algorithm is a symmetric encryption algorithm, based on the idea of sequence cipher, using the key sequence to encrypt each plaintext by bit. The security performance of the sequence cipher depends on the selection of the key sequence [2]. The key sequence needs to be as random as possible to prevent attackers from cracking the key sequence and decrypting the ciphertext. Studies of sequence cryptography are usually focused on the construction of the key sequence, but this algorithm does not design how to generate the key sequence, instead, by means of mutual encryption of data fragments and multi-connection transmission to achieve the effect of encryption. The use of multi-connection and time slot transmission strategy achieves isolation among data fragments transmitted on different connections. For a data fragment, the key sequence used to encrypt it is transmitted through another “logical security channel”, and it is encrypted as ciphertext and transmitted in the current channel. The transmission diagram is shown in Figure 1. As attackers cannot obtain continuous data in the correct order, data on each connection will have a satisfying encryption effect.

2. Related Work

According to the view of modern cryptography, the cryptosystem can be divided into two categories: symmetric cryptosystem and asymmetric cryptosystem. The symmetric cipher includes block cipher and sequence cipher [3]. Block cipher divides plaintext into groups with a specified length, and then encrypts each group with the same key and encryption algorithm into a certain length of ciphertext. The core of block cipher is that by the principle that a simple function iterating several times can turn into a complex function, using simple functions and pairs of operations with non-linear operations realizes encryption [4]. Taking the DES algorithm as an example, its working key is 56-bit, and the encrypted data are 64-bit. The ciphertext output is realized by iterative permutation and inverse permutation of plaintext [5]. There are three main ways to attack the DES system: poor search attack, differential attack and linear attack.

In 1949, Shannon, the founder of information theory [6], proposed that for sequence cipher only "one-time pad" is theoretically unbreakable and absolutely safe. However, except in the case of high confidentiality requirements, the application of "one-time pad" is very limited, because the generation, distribution and transmission of its keys will consume a lot of resources. Nonetheless, due to flexible length, fast operation and low error rate as characteristics of sequence cipher, sequence cipher whose researches focus on pseudo-random sequence as encryption key is still one of the mainstream of current encryption research [7]. The general principle of sequence cipher design [8] is to use multiple keys, multiple calculations and multiple security measures to achieve the similar effect of "one-time pad". The security of sequence cipher depends mainly on the randomness of the encryption key sequence.

With the development of the Internet, secure data transmission become more and more important. There are different works that guarantee information security and increase performance efficiency. Fausto Meneses [9] optimized the RSA encryption algorithm to improve the security, integrity and availability of information. Amit Verma [10] improved the security of the AES by making enhancement in the key matrix of the applied key expansion. Souvik Singha [11] proposed a bit level encryption and decryption algorithm based on the number of keys. The encryption algorithm can encrypt the 8 bit binary number to its corresponding 8 bit cipher text and the decryption algorithm can convert that 8 bit cipher text to its corresponding 8 bit original number. Xin Pu [12] presented a new algorithm which combined the true random sequences and the Tree Parity Machine (TPM). The true random sequences generated by artificial circuits are also proposed as dynamic inputs of TPM compared to the pseudo-random sequences. Though security can be increased to some extent by increasing the length of the key, the efficiency can be reduced only by increasing the key length [13].

Chaotic encryption algorithm represents a new encryption way with less memory consumption and operation time. Feng-Hsiag Hsiao [14] showed a design methodology for neural-network(NN)-based secure communications in multiple time-delay chaotic (MTDC) systems to obtain double encryption via RSA algorithm and chaotic synchronization. Accordingly, it takes much more time to process RSA algorithm if the length of the key is much longer. So a shorter key is used to decrease the processing time of RSA algorithm in this system. However, the method of chaotic cryptography is not yet fully mature [15]. There is some problems remaining to be dealt with for one-dimensional chaotic encryption algorithm, such as small key space, low security and other issues.

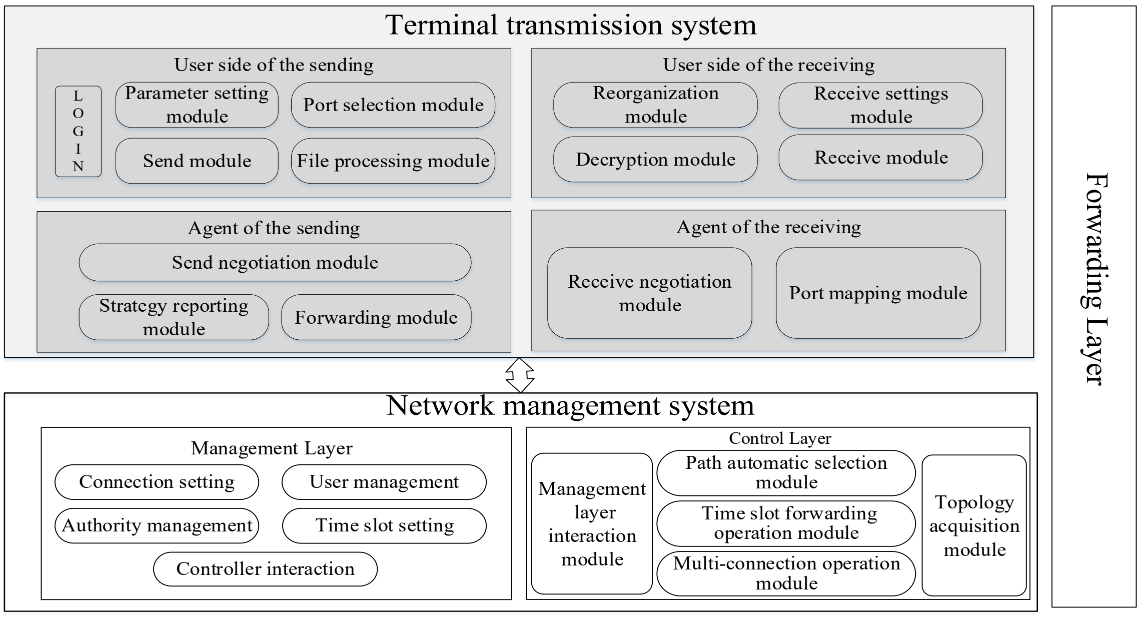

Figure 2. Architecture of the Secure Channel Service System

In order not to increase the amount of data transferred or other reasons, the key sequence will not be too large. As a result, the key sequence is difficult to achieve true randomization. Combined with the advantages of the secure channel service system, this Multi-connection Encryption Algorithm can make the key sequence quite random without sharp increase in transmitting data.

3. System Design

3.1. System Structure

The Multi-Connection Encryption Algorithm is applied in the Secure Channel Service System. The overall architecture of the secure channel service system is shown in Figure 2.

It is divided into two parts: network management system and terminal transmission system. The network management system includes management layer, control layer and forwarding layer. The forwarding layer is an internal network composed of OpenFlow switches. The time-slot forwarding strategy generated by the management layer delivers the flow table to the switch through the control layer. In this way, the switch in the forwarding layer will forward data according to the specified path. The control layer includes a management layer interaction module, a path automatic selection module, a time slot forwarding operation module, a multi-connection operation module, and a topology acquisition module. The management layer includes a connection setting module, a user management module, an encryption setting module, a time slot setting module, an authority management module, a slice setting module, and a controller interaction module. This part provides the service of time slot forwarding strategy for the security channel service, which is transparent to the user.

The terminal transmission system can be divided into four major components according to the deployment position of the security channel service: the user side of the sending, the agent of the sending, the user side of the receiving, and the agent of the receiving. Both the user side of sending and receiving are deployed on the user's terminal hosts. The sender agent and receiver agent can be deployed on the user host, and can also be deployed on a separate proxy server to provide services. These four components are responsible for sending and receiving service processing in different deployment locations.

3.2. Working Mechanism

The process of system normal operation involves user login, negotiation between user side of sending and agent of sending, negotiation between agent of sending and agent of receiving, negotiation between agent of receiving and user side of receiving, sending file processing and receiving file processing, etc. .

The main user behaviours using this system are sending files and receiving files. After both the sender and the receiver have successfully logged in, they can start sending files. After the user of sending selects related parameters in the sender's terminal transmission system and agrees to send, the sender system starts sending related file information (file name, file size, type, etc.), file processing information (file fragmentation size, encryption method) and number of ports for establishing multiple connections to the receiver system.

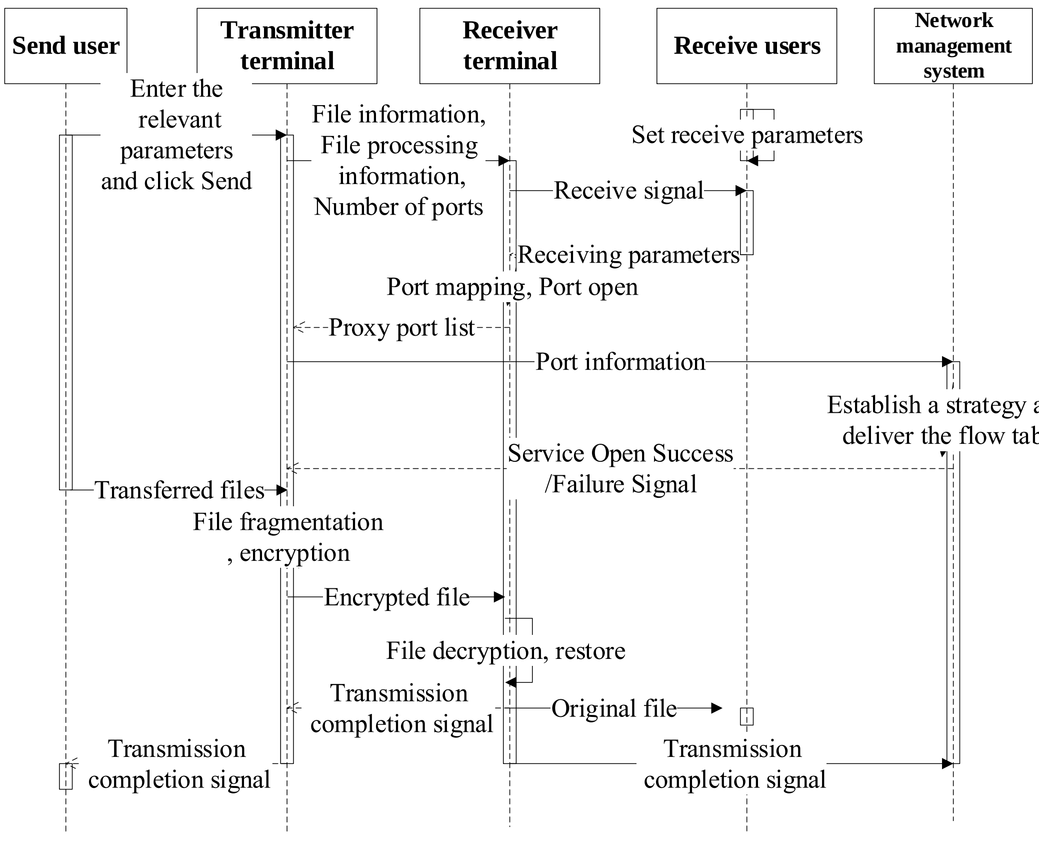
Send user
Transmitter terminal
Receiver terminal
Receive users
Network management system
Enter the relevant parameters and click Send
File information, File processing information, Number of ports
Set receive parameters
Receive signal
Receiving parameters
Port mapping, Port open
Proxy port list
Port information
Establish a strategy and deliver the flow table
Service Open Success /Failure Signal
Transferred files
File fragmentation , encryption
Encrypted file
File decryption, restore
Transmission completion signal
Original file
Transmission completion signal
Transmission completion signal

Figure 3. Sequence Diagram of the Sending Process

After receiving the negotiation request, the receiver sends a receive signal to the receiving user. After the receiving user receives the signal, it queries whether the user modifies the received parameters and starts the receiving port, and then returns the receiving port to the receiving system. After receiving the port list returned by the user, the receiving system maps the proxy port to the receiving port through SSH for file forwarding, and then send the port list back to the sender system. The sender system reports the port list returned by the receiver to the network management system. The network management system establishes the relevant time slot forwarding strategy and sends the strategy to the OpenFlow switch at the forwarding layer through the controller. The network management system returns service opening information to the sender system after starting the service. Then, the sender system performs fragmentation and encryption processing on the file to be sent by the user, and sends it to the receiver through the previously established port. After receiving the file, the receiver system decrypts and reassembles the file, stores the reassembled original file in the path set by the user, and sends a transmission completion signal to the network management system and the sender system. Finally, sender, sending agent, receiving agent, and receiver stops asynchronously. The overall sequence diagram of the security channel service system during the sending process is shown in Figure 3.

4. Algorithm Design

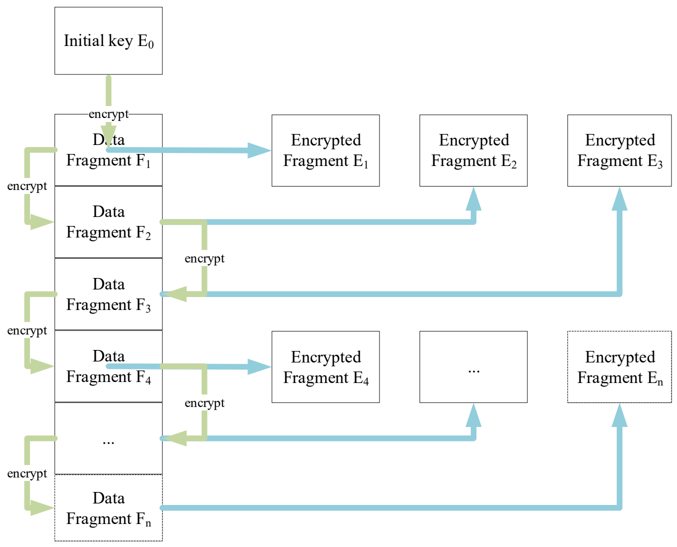

Figure 4. Encryption Process of Data Fragment

```
Algorithm 1: File Processing Algorithm
Input： file,
        length,  // the length of fragment
        encryption, //the method of encryption
        Socket[], //the list of sender port
        connection  //the number of connection
Output：none
Method：
/**Step1：Generate key and send it**/
If encryption == 1
    Use AES to generate key[connection];
    Send key[i] through port
    Socket[(i+1)%connection];
If encryption == 2
    Use IDEA to generate key[connection];
    Send key[i] through port
    Socket[(i+1)%connection];
If encryption == 3
    Use RC5 to generate key[connection];
    Send key[i] through port
    Socket[(i+1)%connection];
If encryption == 4
    Use Multi-Connection to generate key;
    Send key[i] through port Socket[connection-1];

/**Step2：Encrypt data and send**/
While FileInputStream != null
    Read data of length length from FileInputStream
    into Buffer;
    Counter i=0;
    If encryption == 1
        Use key[i%connection] as the key;
        Encrypt Buffer with corresponding encryption
         algorithm;
    else
        Use key;
        Encrypt Buffer with multi-connection
         encryption algorithm;
    i++
    Send Buffer through port Socket[i%connection]
```

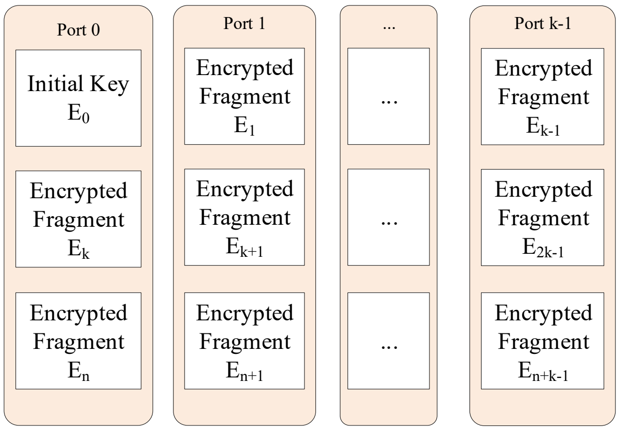

Figure 5. Send Encrypted Fragments through Corresponding Ports

4.1. File Processing, Encryption, Decryption and Multi-connection Transmission

We suppose that the number of TCP connections for transmission is set to be k, data to be transmitted are sliced into n fragments, $F_i \in \{F_1, F_2, \cdots, F_n\}$, initial key sequence is E_0 which has the same length of data fragments, and F_i turns into encrypted fragment E_i after encrypted. When i=0, initial key sequence E_0 is used as encryption key to encrypt fragment F_1 into encrypted fragment E_1, and then E_0 and E_1 will be sent respectively through port $i\%k$. When $1 \leq i < n$, data fragment F_i is used as encryption key to encrypt fragment F_{i+1} into encrypted fragment E_{i+1}, and then E_{i+1} will be sent through port $i\%k$. The file processing algorithm is shown in Algorithm 1. The process of file processing will have some differences based on the selected encryption algorithm, mainly the key generation and transmission process. After the data fragment is processed, it is sent directly and then the next file fragment is continued processing. The encrypting process is showed in Figure 4. The process of sending encrypted fragments through corresponding ports is showed in Figure 5.

Encrypting the data to be transmitted requires that the data be divided into equal-length fragments, and then each fragment is processed in turn as plaintext and the key to encrypt the next fragment. The flow chart is shown in Figure 6. After the length of the data fragment named *length* and the number of connections named *connection* for transmission is determined, the specific process of data encryption and transmission can start.

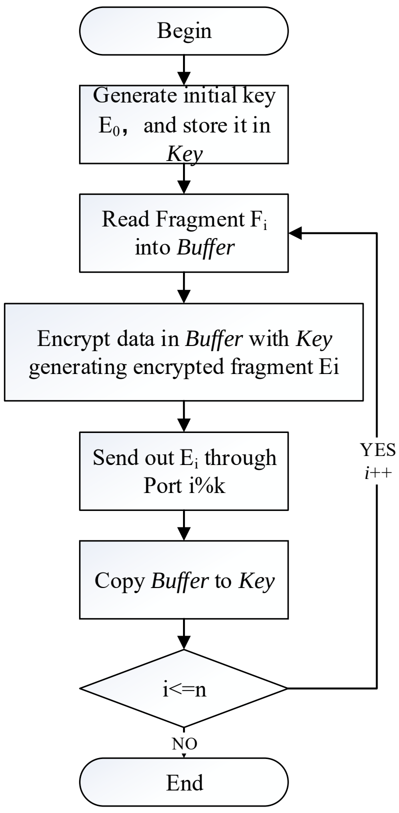
Begin
Generate initial key E0, and store it in Key
Read Fragment Fi into Buffer
Encrypt data in Buffer with Key generating encrypted fragment Ei
Send out Ei through Port i%k
Copy Buffer to Key
i<=n
YES
i++
NO
End

Figure 6. Flow Diagram of Data Encrypted and Sent

Firstly, use the generator to generate a key sequence of length *length*, and store it in the byte array *Key*, and send the *Key* through the port numbered *connection-1* in the port list (the port list contains *connection* port numbers, numbered from 0 to *connection-1*). Secondly, read file data of *length* bytes from the file input stream to the byte array *Buffer*. At the same time, initialize a counter *i* with the value of *0* that is used to judge which port the current fragment should be sent through. Thirdly, use *Key* as the key to encrypt *Buffer* using modulo-2 addition as encryption function, and cache the encrypted data to byte array *Cipher*. Finally, send *Cipher* through the port whose number is *i%connection*. Use *Buffer* as the new *Key* and set *i* with the value of *i+1*. If the file still remains some bytes to read, then repeat the above steps; otherwise, the process ends.

According to the idea of sequence cipher, every bit of plaintext is operated by key sequence, and the operation function uses XOR. x_i is the *i* bit of plaintext, y_i is the *i* bit of ciphertext, s_i is the *i* bit of key sequence (x_i, y_i, $s_i \in \{0,1\}$) The formula of encryption is in (1) and the one of decryption is in (2):

$$y_i = e_{si}(x_i) \equiv (x_i + s_i) \bmod 2 \tag{1}$$

$$x_i = d_{si}(y_i) \equiv (y_i + s_i) \bmod 2 \tag{2}$$

The pseudo-code of encryption and transmission is as described in Algorithm 2.

```
Algorithm 2: Encryption and Transmission
Input: FILE, // file to be transmitted
        k_connections, // number of connections
        SIZE // length of data fragments
Output: none
Method:
  begin
  allocate Key //to store data fragments $F_{i-1}$
  allocate Buffer //to store data fragments $F_i$
  FileInputStream file=new FileInputStream(FILE)
  generate $E_0$ and store it into Key //initial key with length of SIZE
  for (int i = 1;i < FILE.length/SIZE+1; i++){
    Key = file   //read-in next data fragment
    Send (encryption(Key, Buffer))   //send out
    Key.copy(Buffer)
  }
  end
```

The data receiver reads data streams from multiple ports, obtains the length *length* and port number *connection* of the data fragment before receiving, reads and decrypts data fragments sequentially in the order of the port list. The specific process of data receiving and decryption is shown in Figure 7.

Firstly, read *length* bytes of data from the port number list of *connection-1* and buffer it to byte array *Key*, which is the initial key and initialize the counter *i* with the value of 0. Then,

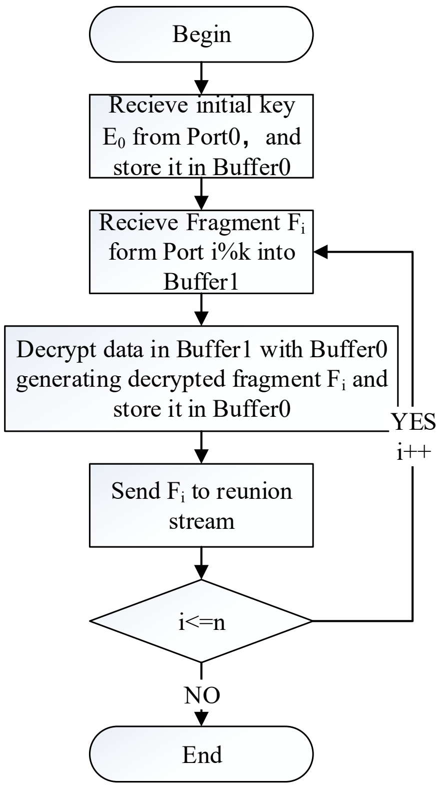
Begin
Recieve initial key E_0 from Port0, and store it in Buffer0
Recieve Fragment F_i form Port i%k into Buffer1
Decrypt data in Buffer1 with Buffer0 generating decrypted fragment F_i and store it in Buffer0
YES
i++
Send F_i to reunion stream
i<=n
NO
End

Figure 7. Flow Diagram of Data Received and Decrypted

from the port of number *i%connection* in the port list read *length* bytes of data, and buffer it to the byte array *Buffer*. Finally, use *Key* as the decryption key to decrypt the *Buffer*,and store it in the buffer byte array *PlainText*. After that, use *PlainText* as the new *Key* and output *PlainText* to the file output stream. If there is data flow still reading in through the port whose number is i%connection, then set counter *i* with value of *i+1* and repeat the steps which is described above; otherwise, the process ends.

4.2. Parameter Setting

- Initial key sequence

The initial key sequence E_0 is only used once during encrypting and decrypting the first data fragment F_1. Its length equals to every data fragment (except for the last one, which may be shorter than others). Since the key sequence does not require large data size, it can be a random number generated by the true random generator (TRNG).

- Size of data fragments

The size of data fragments has direct influence on encryption effect. Because every adjacent data fragment is transmitted on different TCP connection, considering transmission efficiency, to maximum the use of TCP transmission, the size of data fragments should be an integer multiple of TCP MSS (Maximum Segment Size). Sometimes total data size is smaller than MSS. Under this circumstance the first priority is to disperse data to multiple connections, so data fragment size should not beyond the result that total size divides number of connections. Assuming that different connections are completely isolated (attackers can only intercept data on one connection), data on each connection is not what encrypted continuously. If the fragment size is too large, the encryption rule may be exposed [16] [17]: because the encryption key is data itself, different from random sequence it has a certain rule, such as English text encoded with ASCII which has a small key space will be easily cracked when data size is large enough. Therefore, fragment size should be relatively small to ensure the randomness of key sequences, thereby increasing the difficulty of deciphering and thus to guarantee data security.

In conclusion, the total size of transmitting data is SIZE, the number of connections is k, and the size of fragment is decided to be LENGTH=min{MSS,SIZE/k}. When the total size is greater than k*MSS, the fragment size is corresponding to the size TCP maximum message because it will be advantageous for efficient stream processing of splitting, encrypting and transmitting. When data size is less than k*MSS, it should be split according to the number of connections to ensure isolation of adjacent encrypted data. The size of fragment is decided, but the last fragment of data may be less than the size. Instead of filling up to align, it will keep its data size and be encrypted by the previous data fragment with its size, then transmitted.

5. Security and Performance Evaluation

5.1. Comparison of AES and RSA

AES is a symmetric encryption algorithm based on replacement and permutation. The key length can be 128 bits, 192 bits or 256 bits, the block length is 128 bits, and the plaintext block matrix is computed with the key generated by multiple rounds of repeated permutations to achieve encryption [18]. Operation speed of AES encryption is quite fast. The encrypting time and the size of the encrypted data is linearly increased. Encrypted ciphertext size is twice the original data, and thus the decrypting time is twice the encrypting time. The encryption algorithm proposed in this paper does not increase the amount of data except for the initial key that can almost be ignored. For large amount of transmitting data, in comparison with ASE, this algorithm can achieve the effect of encryption with the minimum transmission capacity.

RSA public key encryption algorithm is an asymmetric encryption algorithm [19], based on a special reversible modular exponentiation, of which security is provided by the difficulty of factorial decomposition of the difficulty of large numbers. Take two large numbers with one open to public and another confidential, calculate the encryption key and decryption key, digitize the plaintext and select a certain length of the number as a plaintext block, and use the key to encrypt/decrypt the text/ciphertext [20]. RSA's greatest flaw is that its operation speed is slow. Because RSA requires big numbers computing, the operation speed is 1000 times slower than symmetric cipher algorithms with the corresponding security level. In general, RSA is only used for small amounts of data encryption or encrypting encryption keys. When users' transmission data are large, RSA's slow operation speed makes it not applicable for encrypting all the information. Under the circumstance of SDN Security Transmission Service, all data must be encrypted, so the algorithm in this paper can not only guarantee the speed of encryption and decryption operations, but also use multi-connection to provide better security.

5.2. The Impact of Multi-connections, Network Size and Fragment Size on Security Performance

The time-slot transmission service provided by SDN Security Transmission Service is transparent to users who use encryption and decryption policies, achieving an effect of isolating data that are transmitted on different connections. In general, when the encryption policy is not used, assuming that the number of switches in the entire network is *m*, the number of the alternate path is *s*, the number of switches on each alternate path is *n*, and the number of key switches is l ($l \leq sn$) and assuming that the events for obtaining the switches are independent, the probability of intercepting all data in the case of obtaining one switch is expressed as (3):

$$P_1 = \frac{l}{m} \leq \frac{sn}{m} \tag{3}$$

The probability of intercepting all data in the case of obtaining *k* switches is expressed as (4) and (5):

$$\mathrm{P_k} = 1 - \left(1 - \frac{\mathrm{l}}{\mathrm{m}}\right)^{\mathrm{k}} \qquad \mathrm{k} < \mathrm{s} \tag{4}$$

$$P_k \geq 1 - s \cdot \left(\frac{(s-1) \cdot n}{m}\right)^k \qquad k \geq s \tag{5}$$

In the above situation, in order to ensure that we get all the data, the number of switches we need to obtain on alternate path satisfies the expression as (6):

$$k \geq (s-1)n + 1 \tag{6}$$

The multi-connection service transmits the encrypted data fragments through multiple TCP connections established by multiple ports, and decrypts them at the receiving end. Each connection adopts time-slot forwarding policy to transmit the data independently so they are isolated from each other. Regardless of the encryption policy, if the number of multiple connections is *h* and the network size of every connection is the same as we described above, then the probability of intercepting all the data in the case of obtaining one switch in each connection is expressed as (7):

$$\mathrm{P_1} = \left(\frac{\mathrm{l}}{\mathrm{m}}\right)^{\mathrm{h}} \tag{7}$$

The probability of intercepting all data in the case of obtaining k switches in each connection is expressed as (8) and (9):

$$P_k = \left(1 - \left(1 - \frac{l}{m}\right)^k\right)^h \qquad k < s \qquad (8)$$

$$P_k \geq \left(1 - s \cdot \left(\frac{(s-1)\cdot n}{m}\right)^k\right)^h \qquad k \geq s \qquad (9)$$

In this multi-connection situation, we can guarantee that we get all the data if and only if the number of switches we obtained on alternate path satisfies the expression as (10):

$$k \geq (sh - 1)\cdot n + 1 \qquad (10)$$

Therefore, assuming that the network size is constant, the larger the number of connections is, the less the amount of information being leaked will be when the number of switches acquired is the same. For encrypted data, the less information leaked, the more difficult to decipher its contents by intercepting the information it will be, even if the encryption mode exposes. Without considering the network transmission performance, selecting more switches as alternative paths decrease the possibility of information leakage. This is because the increase in the number of time slots and connections also increases the number of switches used in the network, and the data is more evenly distributed over the network, so the probability of obtaining all the data by obtaining a part of the switches will be much lower. Thus, the number of connections and network size does not directly affect its encryption effect for each encrypted fragment, because each fragment is formed by encryption of the plaintext and the key, and the most important influence factor of encryption effect is the randomness of the key. However, considering the transmission of the entire data, if logically continuous encrypted fragment is captured by attackers, they only need to decipher one part of content, then remaining fragments of the information will be all leaked, so multi-connection and time-slot forwarding provide isolation between adjacent fragments, and the isolation effect is influenced by the number of connections and network size.

5.3. Experiment Method

Implement RSA, DES, AES and the multi-connection encryption algorithm proposed in this paper using JAVA program. Record the execution time of using these four algorithms to encrypt and decrypt text-based data and compare their performance.

In the experiment, a string of 400 bytes is encrypted and decrypted (regardless of the transmission delay, etc.). Each algorithm performs 50 times respectively. The average of the operation time is as follows:

Table 1. Time Consuming of Encrypting and Decrypting a String by Different Algorithms

Algorithms	***DES***	***AES***	***RSA***	***Multi-connection Encryption Algorithm***
Encrypting time (ms)	11.51	43.58	176.09	0.16
Decrypting time (ms)	0.44	0.13	9.67	0.01

The result of time consuming showed in the table above demonstrates that this algorithm has better performance than other encryption algorithms in terms of speed.

The security of the cryptosystem depends on the strength of the security of the algorithm itself, and on the other hand, it is influenced by environmental factors such as the network. The encryption algorithm based on multi-connection transmission belongs to serial cipher encryption. The randomness of the key sequence directly affects the confidentiality of the algorithm. But for an attacker, brute force to crack the cryptosystem is very costly, because the sequence password itself corresponds to the same size of the key sequence. And from the underlying binary sequence, the key does not have regularity. However, according to the coding characteristics of the transmission file itself, the key will also exhibit certain rules. For example, using an ASCII-encoded text file may also show certain rules after being encrypted. In this case we can make the key semantically more random by reducing the size of the data slice, as it is assumed to be in the unit of a TCP message, which includes multiple encrypted data fragments, whose key characteristics are weakened as the fragmentation becomes dense, thus making the attacker harder to infer the rules of the key.

Encrypt an ASCII-encoded text file with 2866 English words using a multi-connection transport-based encryption algorithm. The frequency of appearance of the English letter a(A)-z(Z) in plaintext before encryption is shown in Figure 8. The fragment size is set to 128 bytes (ciphertext 1), 1024 bytes (ciphertext2) and 512 bytes (ciphertext3). After the encryption, the letter frequency analysis is performed. The letter frequency analysis of the three ciphertexts is shown in Figure 9. From this result it can be seen that the frequency characteristics of the text can be completely hidden after encryption. The letter frequency characteristics are roughly similar, but there are also certain differences according to the fragment size; the larger the fragmentation, the greater the difference in frequency between letters, indicating that the fragment size needs may be small. Therefore, the cryptosystem hides the data characteristics of the text file well. The cost of deciphering the cryptosystem under unknown data fragment size and other parameters is relatively high. Combined with the strategy of the security channel service in the network side, we can basically think that the algorithm can meet the high security requirements of users in the security channel service.

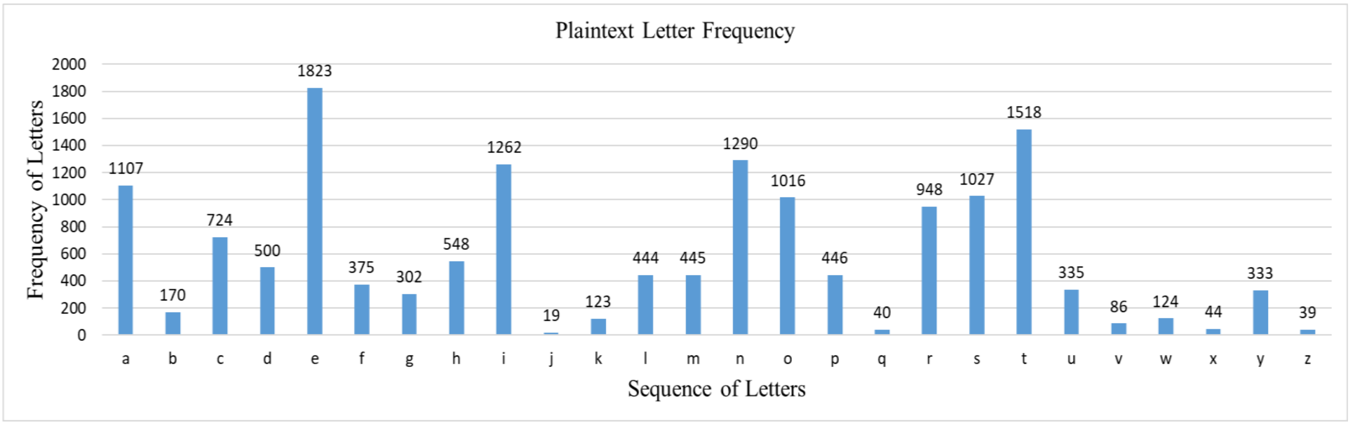

Figure 8. Plaintext Letter Frequency Histogram

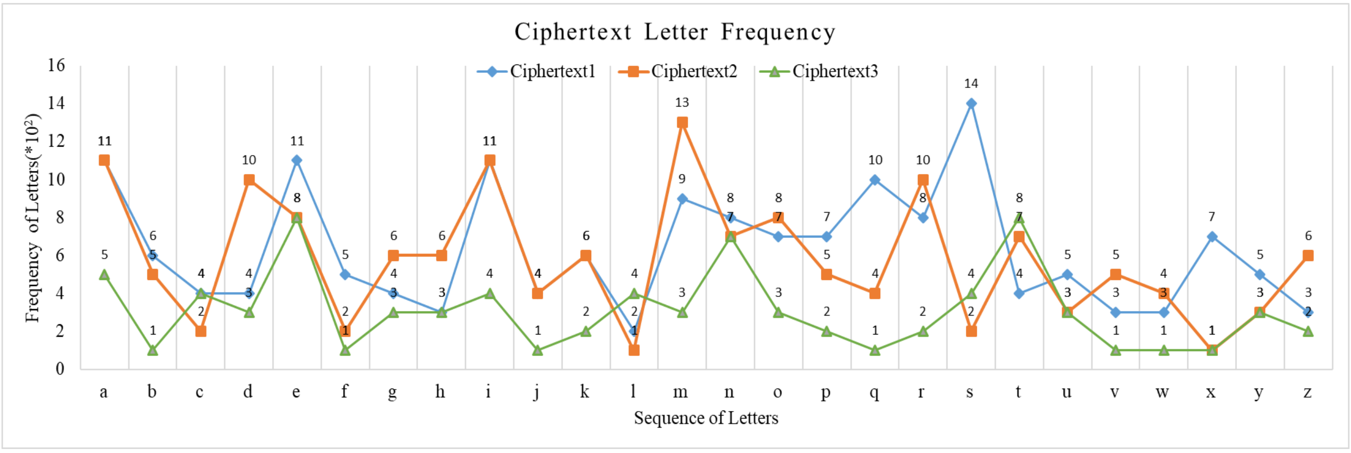

Figure 9. Ciphertext Letter Frequency Comparison Line

6. Conclusion

This paper presents a data encryption strategy in SDN Security Transmission Service. Using the multi-connection and time-slot transmission service provided by SDN Security Transmission Service can isolate the adjacent data fragments. With this feature, the encryption strategy proposed in this paper can guarantee the security of the data transmitted on each connection. Compared with existing encryption algorithms, this algorithm has the advantage of not only preventing the amount of transmission data from largely increasing, but also ensuring the speed of encryption and decryption. Under the condition of multi-connection transmission, the effect of encryption, efficiency and resource utilization achieve the optimal effect and it provides an effective guarantee for network transmission security. Meanwhile, this encryption algorithm is easier to implement and more suitable for such multi-connection scenarios.

Acknowledgements.
This work is supported by the Science and Technology on Communication Networks Laboratory (XX17641X011-06) and Beijing Natural Science Foundation (L171010, 4174099).

References

[1] Stinson D R. “Cryptography: Theory and Practice,” CRC Press, 1995.
[2] Casola V. “Asymmetric Cryptography. Computer Science & Communications Dictionary,” 2007, pp.68-68.
[3] C E Shannon. “A Mathematical Theory of Communication,” Bell System Technical Journal, 1948,27(4), pp.379-423,623-656.
[4] Aoki K, Ichikawa T, Kanda M, et al. “A 128Bit Block Cipher Suitable for Multiple — Platforms Design and Analysis// Selected Areas in Cryptography,” Springer Berlin Heidelberg, 2000, pp.39-56.
[5] Aoki K, Ichikawa T, Kanda M, et al. “A 128-Bit Block Cipher Suitable for Multiple Platforms — Design and Analysis// International Workshop on Selected Areas in Cryptography,” Springer Berlin Heidelberg, 2000, pp.39-56.
[6] C E Shannon. “Communication theory of secrecy systems,” Bell System Technical Journal,1949,28(4) pp.656-715.
[7] Kashmar, Ali H., Ismail, E. S., Hamzah, F. M., and Amir, H. F. A.. "Design a Secure Hybrid Stream Cipher," Journal of Applied Mathematics & Physics, Jun. 2015.

[8] Rueppel R A. “Analysis and Design of Stream Ciphers,” Communications & Control Engineering, 1986.
[9] Meneses F, Fuertes W, Sancho J, et al. RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages[J]. IJCSNS, 2016, 16(8): 55.
[10] Verma A, Kaur S, Chhabra B. Improvement in the Performance and Security of Advanced Encryption Standard Using AES Algorithm and Comparison with Blowfish[J]. 2016.
[11] Singha S, Sen M. Encoding algorithm using bit level encryption and decryption technique[C]//Computer, Electrical & Communication Engineering (ICCECE), 2016 International Conference on. IEEE, 2016: 1-4.
[12] Pu X, Tian X, Zhang J, et al. Chaotic multimedia stream cipher scheme based on true random sequence combined with tree parity machine[J]. Multimedia Tools and Applications, 2017, 76(19): 19881-19895.
[13] Jia N, Liu S, Ding Q, et al. A New Method of Encryption Algorithm Based on Chaos and ECC[J]. Journal of Information Hiding and Multimedia Signal Processing, 2016, 7(3).
[14] Hsiao F H. Chaotic synchronization cryptosystems combined with RSA encryption algorithm[J]. Fuzzy Sets and Systems, 2017.
[15] Liu S. Research on the design and implementation of two dimensional hyper chaotic sequence cipher algorithm[C]//Future Generation Communication Technologies (FGCT), 2017 Sixth International Conference on. IEEE, 2017: 1-4.
[16] Meier W, Staffelbach O. “Fast correlation attacks on stream ciphers,” The Workshop on Advances in Cryptology-Eurocrypt. Springer-Verlag New York, Inc. 1988, pp.301-314.
[17] Fidus, Felsa Mary, K. S. Lalmohan, and A. Sekhar., "Design and Implementation of a Secure Stream Cipher for Cryptographic Applications." International Journal of Engineering & Technical Research V4.7, 2015.
[18] Daemen J, Rijmen V. “The design of Rijndael: AES, the advanced Encryption Standard,” 2002.
[19] Rivest R, Shamir A, Aldeman L. “A method for obtaining digital signatures and public-key cryptosystems,” Communications of the ACM, 1978, 21(2), pp.120-126.
[20] Lin, Xi Jun, L. Sun, and H. Qu, “An efficient RSA-based certificateless public key encryption scheme,” Discrete Applied Mathematics, Mar. 2017, doi:10.101.