Performance of Transport Layer Protocals for Secure Data Transaction

Mr. M. Arumai Selvam
HOD,
PG and Research Department of Computer Science
St.Joseph's College of Arts and Science (Autonomous)
Cuddalore, Tamil Nadu.

C. Victor Vimal Raj
M.Phil Scholar,
PG and Research Department of Computer Science
St.Joseph's College of Arts and Science (Autonomous)
Cuddalore, Tamil Nadu.

R. Sachidhanandham
M.Phil Scholar,
PG and Research Department of Computer Science
St.Joseph's College of Arts and Science (Autonomous)
Cuddalore, Tamil Nadu.

R. Sangeetha
M.Phil Scholar,
PG and Research Department of Computer Science
St.Joseph's College of Arts and Science (Autonomous)
Cuddalore, Tamil Nadu.

***Abstract* - The transport layer protocols are most used in computer networks. The transport layer is the heart of the whole protocol hierarchy. Protocols are used to transferring the data between two hosts through the internet. In this paper will discuss on combining the two protocols they are transmission control protocol and User datagram protocol. The TCP where loss of a single data unit it makes the hole data transfer unless such as file Transfer, so the TCP is reliability. UDP is unreliable and can be used where little loss is acceptable and sender does not need a confirmation of successful delivery to the receiver. In this paper discuss on incensing the performance and security without affecting the complexity level on client and server side.**

Keywords: Data Transaction, Security, Layer, TCP, UDP

1. INTRODUCTION

The data can be transmitted various protocols here two protocols mainly focused. First one is TCP for reliability and second one is UDP for fastness of data transaction. The UDP does not secure and does not guarantee the deliver the packet. TCP does not fast and secured. So combining the two protocols and transmit the data. Now a day the security has become an impartment issue, so the data security has become a basic necessity. In this paper involves the security of data through 6 layers[1].The data will pass the 6 layers of security on the sender's side. The first layer defines compress of any encryption algorithms[2]. The encryption service is impartment because it is needed for transmission of information and authentication for verify excess level of user. The second layer involves compression technique. The third layer involves a new technique of key generation. The forth layer the data traversed in any one of the 9 existing patterns. The fifth layer adds the data with any one of text, audio, video, image or file. The last layer combines both UDP and TCP. The byte array of this file will be send through the UDP. After key generated and sent it through the TCP. The transmission of data and Key in different protocols and improve the security to a most extend. Improvement will be increasing the client and server side without affecting the complexity level and [2]increase the performance and security of transaction and generate dynamic key and parallel encryption and decryption.

2. PROPOSED WORK

From the last year various protocols have been developed. Among them various connectionless and connection protocol provide data transmission over the network. Mainly the data transfer using the two protocols they are UDP and TCP for implement the snake horizontal traversal[1] or any other traversal. This technique can be extending by implementing all 9 varieties of traversals in such a way that the traversal to be performed has to be selected at random as the security increases. Similarly, this paper has been implemented by using the any longer word for key generation.

3. EXPERIMENT

First select the file or data that has to be transmitted to the destination. This file or data has to be encrypted and compressed, which forms the first 2 layers of security to be implemented.

3.1 ENCRYPTION AND DECRYPTION

There is huge number of encryption and decryption algorithm available in computer science. In this paper is implemented by RSA encryption and decryption algorithm.
Encryption: $F(m, e) = m^e \bmod n = c$, where m is the message, e is the public key and c is the cipher.
Decryption: $F(c, d) = c^d \bmod n = m$.
First generate the public key. It has two public keys that is P and Q. the public key must be a prime number.
$n = P \times Q$
We have to consider two prime numbers. 53 and 59 that is n=3127.
Also we have to one exponent e. the e must be, Be an integer, not be a factor of n and $1<e<\varphi(n)$.
Example: 3.
Our public key is made up of n and e.
n=3127 and e=3.

And second generate the private key. To calculate the φ(n).
$\varphi(n)=(P-1)(Q-1)$
$\varphi(n)=(53-1)(59-1)$
$\varphi(n)=3016$.
Now calculate the private key d.
$d=\frac{2(\varphi(n)+1)}{e}$
$d = \frac{2(3016+1)}{3}$
d = 2011
Now the public key is n(3127) and e(3), and the private key is 2011.
Example:
Let's encrypts the HI,
Converts the Text into Number,
H-8, I-9,
C is the encrypt data.
$C = DATA^e \bmod n$
$C=89^3 \bmod 3127$
C=1394.
Next decrypt the Data with Encrypted data c, public key n and private key d.
The decrypt the data is equal to the encrypt data,
Decrypt data=$c^d \bmod n$
Data=$1394^{2011} \bmod 3127$
Result is 89.

3.2 COMPRESSION AND DECOMPRESSION

The compression of reducing the volume of data by applying a compression technique a compression technique is called compression. The resulting data is called compressed data. The reverse process of reproducing the original data from compressed data is called decompression.
Compression is based on the average value concept. The ASCII values of all characters are obtained, and then each obtained value will be subtracted from a constant value 97.Average of first 2 values will be calculated andthe corresponding character will be replaced from the character set. Similarly, all characters will be replaced by the character corresponding to its average value. All the substituted characters are appended together to get the final compressed text. If the length of the original text is odd then the last left character can be appended directly with the compressed string[1].For example, consider a word "configuration" which will be compressed to "ijhtklh". The complete flow of the process is defined with the same example.
First the ASCII value of each alphabet in the word compress. c- 99, o- 111, n- 110, f- 102, i- 105, g- 103, u- 117, r- 114, a- 97, t- 116, i- 105, o- 111, n- 110. Subtract the ASCII value with a constant 97. So the obtained values will be 21413586201701981413.
Then calculate the average of the neighboring 2 alphabets. If the length is an odd value then use the last alphabet directly.
2+14=16/2=8
13+5=18/2=9
8+6=14/2=7
20+17=37/2= ceil (18.5) = 19
0+19=19/2= ceil (9.5) = 10
8+14=22/2=11
13=13/2= ceil (6.5) = 7
Substitute the character corresponding to the obtained value 8- I, 9- j, 7- h, 19- t, 10- k, 11- l, 7- h.
So the compressed text is "ijhtklh".
The client side is decompression will be done, it happens prior to decryption. The decompression is define by mathematical concept that is A+B=C and A-B=C. During compression process the difference between the neighboring characters will be obtained[3]. Those values will be used to perform decompression. Consider the above example compressed text "ijhtklh". During Compression the difference between neighboring values are as below,
c – o = 2 – 14 = abs (-12) = 12
n – f = 13 - 5 = 8
i – g = 8 – 6 = 2
u – r = 20 - 17= 3
a – t = 0 – 19 = abs (-19) = 19
i – o = 8 – 14 = abs(-6) = 6
n =13
On the client side, the respective number for each alphabet in the received compressed text will be multiplied by 2. This manipulation is similar to finding the sum of the neighboring characters. The received text will be "ijhtklh" in our case. So,
i – 8 * 2 = 16
j – 9 * 2 = 18
h – 7 * 2 = 14
t – 19 * 2 = 38
k – 10 * 2 = 20
l – 11 * 2 = 22
h – 7 * 2 = 14
Above obtained results will be similar to
o + c= 14 + 2 = 16
f + n = 5 + 13 = 18
g + i = 6 + 8 = 14
r + u = 17 + 20 = 37 + 1 = 38
t + a = 19 + 0 = 19+1 = 20
o + I = 14 + 8 = 22
n= 13 +1 = 14
The differences will be used to generate key (will see in detail at key generation). So these differences can be extracted from the key in client side and can be used to find the original characters. In this example on client side values 12, 8, 2, 3, 19, 6 and 13 will be extracted and values 16, 18, 14, 37, 19, 22 and 13 will be calculated.

3.3 KEY GENERATION

In this key generation process, key based on the user key enter for the encryption and the original data will be the encrypted. In this output of the key send it through the TCP connection. The key must always be protected from modification of encrypted files. For the cipher text to be transformed to plaintext, the decryption function must use the same key used by the encryption function to decrypt the cipher text.
Let us consider the same encrypted letter "ijhtklh" example in which length of the compressed text is 7. If consider the same public key a and b like 53 and 59. It must be the prime number. Its 6 bit format of key is 110101 and 111011. As the number of bits in each partition is calculated to be 4, the above appended key should be split up into 3 partitions with maximum 4 bits each. So the segmented key will be as given below 1101 0111 1011. In between each segmented key the

difference calculated in the compression part should be embedded[1]. The differences calculated in compression part are 12, 8, 2, 3, 19, 6 and 13. The binary representation of all these values should be embedded in between each partition. In some cases the differences may be a negative value. So if the value is negative then add 1 prior to its binary representation else add 0. In the above case 12, 19 and 6 will be negative values. The final binary representation of differences will be 101100, 110011, 100110. These differences will be embedded between the key partitions. The final key generated by this algorithm – 10110000100000001000011001100100110101101 [1][4][5].This key sends the user to the client. In this key even the user can not identify the key. On the client side the key will be received. With the help of the differences extracted decompression can be executed. With the original key extracted decryption can be done to get the original message.

3.4 TRAVERSAL

The next layer is 4th its security of traversal, through any traversal. It has 9 different shapes of traversals which are known as snake lake horizontal, snake lake vertical, raster horizontal, raster vertical, z-horizontal, z-vertical, spiral, zigzag and diagonal. Data can be read in either of the above traversal methodology. So the data will be arranged in a different manner from the original manner. The common direction of the traversals is listed here.

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

• Snake Horizontal: A B C D E F G H I J K L M N O P
• Snake Vertical: A E I M B F J N C G K O D H L P

To make still more complex and secure random selection of traversal can be chosen and the respective reverse traversal can be executed on the other end. This random selection will make even the sender unknown about the traversal done and it will not be easy to predict the traversal undergone by any attacker as the data will be misarranged, encrypted and compressed[1]. Consider the word "CONFIGURATION". We can consider 4*4 matrixes so compress in 4*4matrixes will be arranged like

C	O	N	F
I	G	U	R
A	T	I	O
N			

After the traversal data, the result will look like "CONFIGURATION". The dimension of the matrix can be decided by the sender. If that too is made to be random then the increase in security will be directly proportional to the complexity to perform cryptanalysis. On the client side this data should be arranged back to the original order in prior to Key extraction, decompress and decryption[4].

3.5 HIDING DATA

The next layer appended the data with text, image or multimedia file. After appending the data and send it to the client or another end. The client side the file is received but the text, image or multimedia files need not to be save as the client side because no use of this text or another append file. So the appended text can just be extracted and the remaining contents of the file can be ignored. The data will be read from the text/multimedia file and will be passed to traversal module in the client side[1]. As the UDP can support only 60KB (approximately) large multimedia files need to be segmented into packets on server side and integrated on client side. We append the data into a multimedia file so obviously the size of the file to be transmitted will be above 60KB which leads to segmentation[1][4]. In general, Even if one of the packets get lost or damaged the integration of file wouldn't be successful.

3.6 TCP AND UDP TOGETHER

In this section implement security purpose using the TCP and UDP technique. The data to be passed will be passed through the UDP and generated key will be passed through the TCP. Here, the data is the multimedia file or text file to which the traversed data is appended. This file will be segmented into packets and each packet will be transferred through Datagram Sockets. The generated key will be sent it through TCP socket. For example the key generated in above key generation technique is 10110000100000001000011001100100110101101 will be transferred through the TCP connection. This technique of combining both the connections will make the data high secure to transfer from source to destination.
Here, Let us consider 3 cases

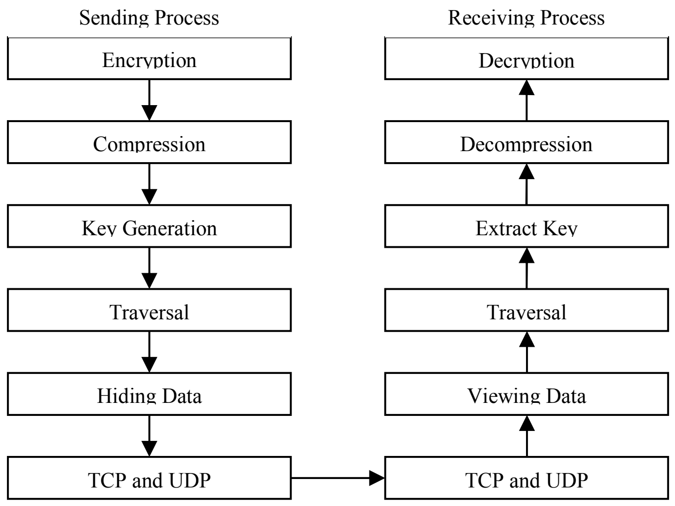
Case 1: If a cryptanalyst attacks the TCP socket, the generated key will alone be obtained by the attacker. To get the original key from this generated key the differences should be extracted. To extract the differences, length of the compressed text should be known from which the number of partitions and number of bits in each partition has to be calculated[6]. But the compressed text is traversed and hidden into a multimedia file which is further being transmitted in UDP (i.e.) in a different connection.

Case 2: If a cryptanalyst attacks the UDP socket, the image or audio file contents will be obtained by the attacker. Here, in this paper the byte array of the image or audio file is only transferred by the UDP socket so as the contents will look like combining the special characters, characters and numerals it will be difficult to analyze that only the appended data will be the original data[1]. If the original data has to be obtained by the cryptanalyst then the position of the appended data should be identified, then it should be traversed in a particular pattern used (pattern will be unknown), decompression should be performed, original key has to be obtained to perform encryption but the key will be passed through TCP connection. So with just the message from the socket without the key it is impossible for the cryptanalyst to read the message.

Case 3: Even if both the channels are being observed then it would be difficult for the cryptanalyst to extract the key from the duplicate key and even all the 6 layers has to be performed in reverse order on the entire data to get back the original data. This reduces the probability level to obtain the real message getting transferred.

These 3 cases, combining TCP and UDP the data can be passed in a secure data and even if any loss of packet occur it does not affect the flow or execution of the process as only the appended data is required.

3.6.1 FLOW DIAGRAM



3.6.2 PERFORMANCE ANALYSIS

TABLE 1

Module	**1000 Characters**	**1500 Characters**
Encryption	20ms	20ms
Compression	1ms	1ms
Decompression	1ms	1ms
Decryption	20ms	20ms
Key Generation	1ms	1ms
UDP Time	4s(approx)	6s(approx)

4. CONCLUSION

In this paper, each layer individually performs in an efficient manner. After integration, the performance level has no degradation and in fact, it imparts better security too. This idea is being enhanced to meet further requirements too. Improvements will be made refer to increasing the performance and security without affecting the complexity level on client and server side. Region based segmentation can be used to hide the data in a particular part of the image. The implementation of these improvements is in process, to enhance the security of the data.

REFERENCES

[1] K. Rajkumar and P. Swaminathan, “Combining TCP and UDP for Secure Data Transfer,” vol. 8, no. May, pp. 285–291, 2015.

[2] M. Abhinivesh, M. Garg, and D. P. Acharjya, “Secured Transaction for Distributed Service System,” vol. 8, no. January, pp. 160–164, 2015.

[3] P. Patel, R. Shah, C. Patel, and V. Vijayarajan, “Reliable Connectionless Transport Protocol for Fast Message Delivery,” vol. 8, no. January, pp. 284–290, 2015.

[4] M. Pratim Sarma, “Performance Measurement of TCP and UDP Using Different Queuing Algorithm in High Speed Local Area Network,” *Int. J. Futur. Comput. Commun.*, vol. 2, no. 6, pp. 682–686, 2013.

[5] A. N. Naqvi, A. M. Abbas, and T. A. Chouhan, “A Performance Evaluation Of IEEE 802.16e Networks For TCP And UDP Traffics,” vol. 1, no. 8, pp. 1–8, 2012.

[6] M. Anbar, S. Ramadass, S. Manickam, and A. Al-wardi, “Connection Failure Message-based Approach for Detecting Sequential and Random TCP Scanning,” vol. 7, no. May, pp. 628–636, 2014.