Security Enhanced Communication Scheme with Compression on Encrypted Cipher Block

Asha R Pillai[1], **Chithra M**[2]

[1]P G Scholar, VLSI and Embedded Systems, Department of ECE, T K M Institute of Technology, Kollam, India

[2]Assistant Professor, Department of ECE, T K M Institute of Technology, Kollam, India

Abstract*: The process of exchanging ideas or information between two persons or stations or devices is termed as communication. The information security has become one of the most significant problem in data communication. The intruder is an unwanted person who reads and changes the information while transmission occurs. So it becomes an inseparable problem in data communication. In order to address this problem, cryptography is used. Cryptography is the science of keeping message secret. It deals with transformations of a message into coded form by encryption at the sender (transmitting) side and recovery of original message by decryption at the receiver side. The classical way of transmitting data over an insecure channel is to first compress it and then encrypt. At the receiver, the received bit stream is first decrypted and then decompressed. But when the data is first compressed, there may be a problem of data loss and also if the sender is not able to perform the compression operation first then it will affect the smooth communication between the sender and the receiver. In order to overcome this problem, the existing technique is reversing such that the data will be first encrypted and then compressed without compromising the information secrecy. When the data is first encrypted with a secret key, then the data become secure and the sender can use the help of a third person to compress the data if the sender is not able to perform the compression operation first. The data security can be achieved using encryption and decryption algorithms. Here the encryption and decryption is performed using the Triple DES algorithm. The Triple DES algorithm is same as that of DES algorithm, in which the DES algorithm is applied three times to each block. The Run length encoding (RLE) method is used for data compression and decompression. The VHDL language is used for coding, synthesis can be done by means of Xilinx ISE and Model Sim can be used for simulation.*

Keywords: Data communication, Intruder, Cryptography, DES algorithm, TDES algorithm, RLE.

1. Introduction

The world is becoming more interconnected with the advent of the Internet and new networking technology. There is a large amount of personal, commercial, military, and government information on networking infrastructures worldwide. Network security is becoming of great importance because of intellectual property that can be easily acquired through the internet. By increasing network security, it will decrease the chance of privacy spoofing, identity or information theft and so on. Network attacks are often caused by direct or indirect interaction of humans. There are many situations in which employees themselves pose the biggest threat to enterprises. Many times, employees will unintentionally install piracy software that is infected with viruses, worms or trojans. Other times, users may forget to secure their workstations, leaving them open as an easy target to potential attackers. And yet others may give sensitive information to outsiders, or even play a role in an important part of an attack.

The security of data transmission is also a vital problem in communication networks. A communication system is reliable as long as it provides high level of security. Usually, users exchange personal information or important documents. In this case the security, integrity, authenticity and confidentiality of the exchanged data should be provided over the transmission medium. Nowadays, internet is very popular in which a significant amount of data is exchanged every second over a non secured channel, which may not be safe. Therefore it is essential to protect the data from attackers. In order to address this problem, cryptography is used. Cryptography is the science of keeping data transfer secure, so that eavesdroppers or attackers cannot decipher the transmitted message. The word cryptography has come from a Greek word, which means secret writing. In the present day context it refers to the tools and techniques used to make messages secure for communication between the participants and make messages immune to attacks by hackers. For private communication through public network, cryptography plays a very crucial role. The role of cryptography can be illustrated with the help of a simple model of cryptography as shown in figure 1.

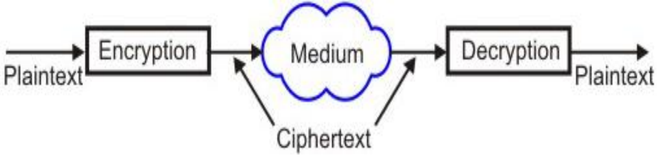

Figure 1: Cryptography model

The message to be sent through an unreliable medium is known as plain text, which is encrypted before sending over the medium. The encrypted message is known as cipher text, which is received at the other end of the medium and decrypted to get back the original plain text message [2].

2. Theory

The variable attacks that affect the network security can be classified as interception, fabrication, modification, and interruption [11]. In order to protect from all this attack there are four chief principles of security that the cryptography offered. They are confidentiality, authentication, integrity, non repudiation, access control, availability. The principle of confidentiality specifies that only the sender and the intended receiver should be able to access the contents of the message.

The interception attack causes loss of message confidentiality. The authentication process ensures that the origin of an electronic message or document is correctly identified. The fabrication attack is possible in the absence of proper authentication mechanisms. If the contents of a message changes after the sender sends it, but before it reaches the intended recipient, then the integrity of the message is lost. The modification attack causes loss of message integrity. Non repudiation does not allow the sender of a message to refute the claim of not sending that message. The access control mechanism specifies and controls who can access what.

In cryptography, the original message is called plain text, when encrypted gives cipher text. The encryption process is defined over the usage of key. In computer to computer communications, the computer at the sender's end usually transforms a plain text message into cipher text by performing encryption. The encrypted cipher text message is then sent to the receiver over a network (such as the internet). The receiver's computer then takes the encrypted message, and performs the reverse of encryption, i.e. it performs the decryption process to obtain the original plain text message. To encrypt a plain text message, the sender performs encryption, i.e. applies the encryption algorithm [6]. To decrypt a received encrypted message, the recipient performs decryption, i.e. applies the decryption algorithm. The second aspect of performing encryption and decryption of message is the key. In general, the algorithm used for encryption and decryption processes is usually known to everybody. However, it is the key used for encryption and decryption that makes the process of cryptography secure.

Broadly there are two cryptographic mechanisms, depending on what keys are used [9]. When a single key is shared between the two entities of the communication, it is called symmetric key cryptography. The DES (Data Encryption Standard) algorithm is an example of symmetric key cryptography. In asymmetric key cryptography, sender encrypts plain text or message using a public key. The receiver then uses their own private key to decrypt the message. In this cryptographic technique public key is known to all, but private key is not shared by the participants [7]. The RSA (Rivest-Shamir-Adleman) algorithm is an example for asymmetric key cryptography. The generation of cipher text from plain text itself can be done in two basic ways, stream ciphers and block ciphers. Stream cipher technique involves the encryption of one plain text bit at a time. The decryption also happens one bit at a time. The block cipher technique involves encryption of one block of text at a time. Decryption also takes one block of encrypted text at a time [4].

3. Methodology

The basic block diagram for the secure transmission of data is shown in figure 2.

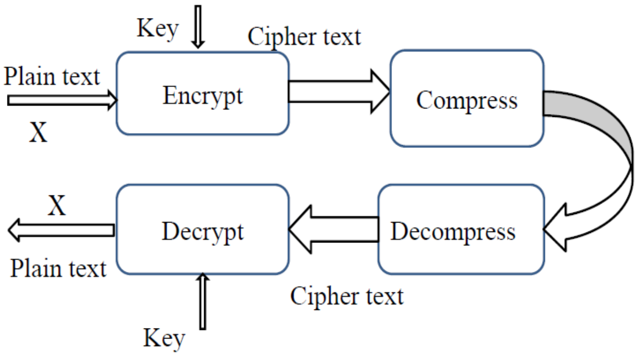

Figure 2: Block diagram

At the sender side, the plain text or the original message is encrypted using triple DES (Data Encryption Standard) algorithm to form the cipher text. The cipher text is then compressed with the run length encoding (RLE) method. The compressed cipher text is then transmitted through the channel. At the receiving side, the compressed cipher text is first decompressed using the same RLE and then decrypted using the triple DES algorithm to retrieve the original plain text or the message.

3.1 Encryption and Decryption using triple DES algorithm

The Triple Data Encryption Standard (TDES) is a block cipher operating on 64-bit data blocks [5]. The combined key size is 168 bits. The key size of DES is 56 bits, therefore the key size of TDES is three times 56 (56*3=168). The triple DES algorithm is same as that of DES algorithm in which the DES is applied three times on a plain text with three different keys. The plain text block P is first encrypted with a key K1, then encrypted with second key K2, and finally with third key K3, where K1, K2 and K3 are different from each other as shown in figure 3. To decrypt the cipher text C and to obtain the plain text, P = DK3(DK2(DK1(C))) operation has to be done where D denotes the decryption, C denotes the cipher text, and K denotes the key [8].

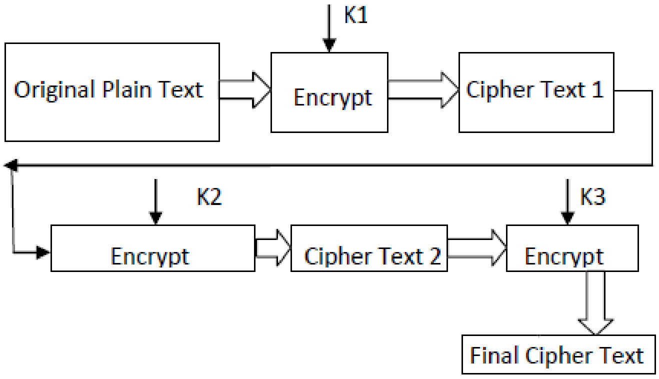

Figure 3: Triple DES

3.2 How DES works

3.2.1 Basic principle

DES is a block cipher. It encrypts data in blocks of size 64 bits each. That is, 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key is used for encryption and decryption. The key length is 56 bits. Actually, the initial key consists of 64 bits. However, before the DES process even starts every eighth bit of the key is discarded to produce a 56 bit key. That is, bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded. Thus the discarding of every eighth bit of the key produces a 56 bit key from the original 64 bit key.

3.2.2 Broad level steps in DES

The broad level steps in DES are shown in figure 4.

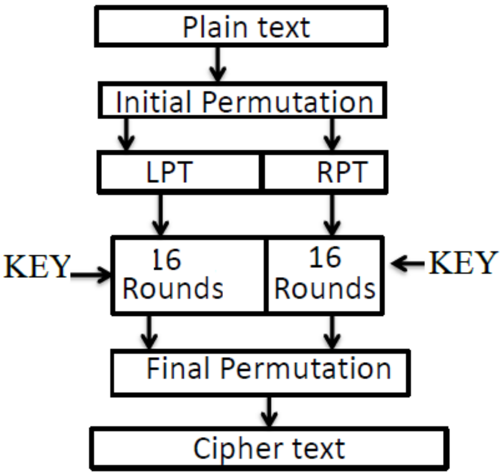

Figure 4: Broad level steps in DES

- In the first step, the 64 bit plain text block is handed over to an initial permutation (IP) function.
- The initial permutation is performed on plain text.
- Next the initial permutation produces two halves of the permuted block; left plain text (LPT) and right plain text (RPT).
- Now each of LPT and RPT goes through 16 rounds of encryption process, each with its own key.
- In the end LPT and RPT are rejoined and a final permutation (FP) is performed on the combined block.
- The result of this process produces 64 bit cipher text.

3.2.2.1 Initial permutation (IP)

The initial permutation (IP) happens only once and it happens before the first round. The initial permutation replaces the first bit of the original plain text block with the 58th bit of the original plain text block, the second bit with the 50th bit of the original plain text block, and so on as shown in the table 1. This is nothing but jigglery of bit positions of the original plain text block to increase the data security.

Table 1: Initial permutation

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

After initial permutation is done, the resulting 64 bit permuted text block is divided into two half blocks. That is left plain text (LPT) and right plain text (RPT). Each half block consists of 32 bits. Then 16 rounds are performed on these two blocks of the original plain text block.

3.2.2.2 Rounds

One round in DES consists of following steps:

Step 1: Key transformation

For each round, a 56 bit key is available. From this 56 bit key, a different 48 bit sub-key is generated during each round using a process called as key transformation. For this, the 56 bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round. For example, if the round number is 1, 2, 9 or 16, the shift is done by only one position. For other rounds, the circular shift is done by two positions.

After an appropriate shift, bit number 14 moves into the first position, bit number 17 moves into the second position and so on as shown in the table 2. Also it contains only 48 bit positions because some bits will be discarded to reduce the 56 bit key to a 48 bit key. Since the key transformation process involves permutation as well as selection of a 48 bit sub set of the original 56 bit key, it is called as compression permutation.

Table 2: Compression permutation

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Because of this compression permutation technique, a different subset of key bits gets used in each round. That makes DES not so easy to crack.

Step 2: Expansion permutation

During expansion permutation, the right plain text is expanded from 32 bits to 48 bits. Besides increasing the bit size from 32 to 48, the bits are permuted as well, hence the name expansion permutation. This happens as follows:

- The 32 bit right plain text is divided into 8 blocks with each block consisting of 4 bits. This is shown in figure 5.

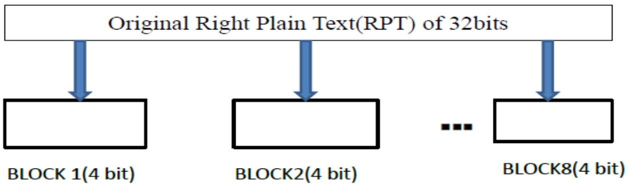

Figure 5: Division of 32 bit RPT into eight 4 bit blocks

- Next, each 4 bit block of the above step is then expanded to a corresponding 6bit block. That is, per 4 bit block, 2 more bits are added. They are actually the repeated first and the fourth bits of the 4 bit block. The second and third bits are written down as they were in the input. This is shown in figure 6. The first input bit is the output to the second output position, and also repeats in output position 48. Similarly, the 32nd input bit is found in the 47th output position as well as in the first output position. This process results into expansion as well as permutation of the input bits while creating the output.

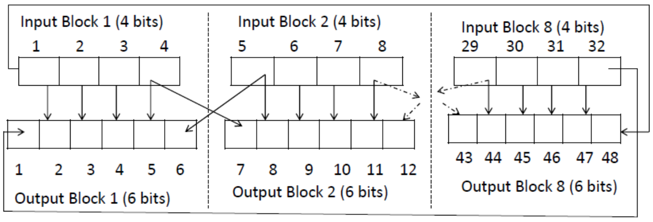

Figure 6: Expansion permutation process

After expansion permutation, the first input bit goes into the second and the 48th output positions. The second input bit goes into the third output position, and so on as shown in the table 3.

Table 3: Expansion permutation

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

As the key transformation process compresses the 56 bit key to 48 bits. The expansion permutation process expands the 32 bit right plain text to 48 bits. Now, the 48 bit key is XORed with the 48 bit RPT, and the resulting output is given to the next step, which is the S box substitution.

Step 3: S box substitution

S box substitution is a process that accepts the 48 bit input from the XOR operation involving the compressed key and expanded RPT, and produces a 32 bit output using the substitution technique. The substitution is performed by eight substitution boxes (also called as S- boxes). Each of the eight S boxes has a 6 bit input and a 4 bit output. The 48 bit input block is divided into 8 sub blocks (each containing 6 bits), and each such sub block is given to a S box. The S box transforms the 6 bit input into a 4 bit output as shown in the figure 7.

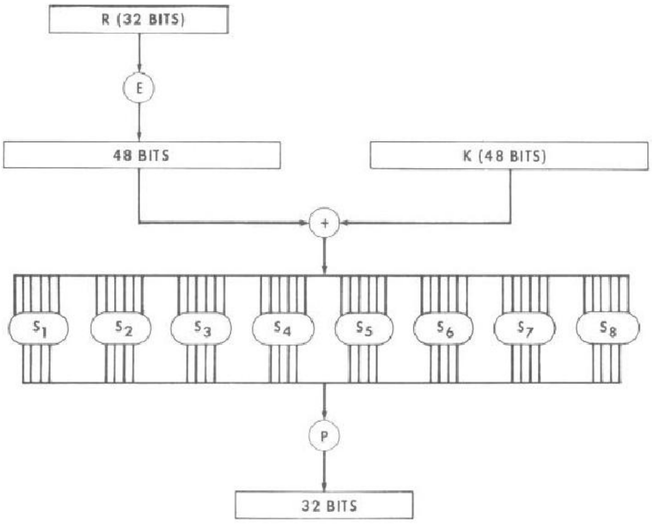

Figure 7: S box substitution

The logic used by S box substitution for selecting only four of the six bits is that assume every S box as a table that has 4 rows and 16 columns. Thus there are 8 such tables, one for each S box. This is shown in table 4 to table 11. At the intersection of every row and column, a 4 bit number (which will be the 4 bit output for that S box) is present. The 6 bit input indicates which row and column, and therefore, which intersection is to be selected, and thus determines the 4 bit output.

Table 4: S box 1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Table 5: S box 2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Table 6: S box 3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Table 7: S box 4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Table 8: S box 5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Table 9: S box 6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Table 10: S box 7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Table 11: S box 8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

For example assume that the six bits of a S box are indicated by b1, b2, b3, b4, b5 and b6. Now bits b1 and b6 are combined to form a two bit number. Two bits can store any decimal number between 0 (binary 00) and 3 (binary 11). This specifies the row number. The remaining four bits b2, b3, b4, b5 make up a four bit number, which specifies the column number between decimal 0 (binary 0000) and 15 (binary 1111). Thus, the 6 bit input automatically selects the row number and column number for the selection of the output. This is shown in figure 8.

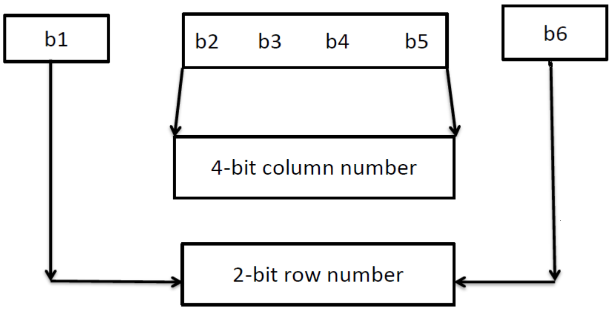

Figure 8: Selecting an entry in a S box based on the 6 bit input

Suppose the 7 to 12 of the 48 bit input (that is the input to the second S box) contain a value 101101 in binary. Therefore, we have (b1,b6) = 11 in binary (i.e. 3 in decimal), and (b2, b3, b4, b5) = 0110 in binary (i.e. 6 in decimal). Thus the output of S box 2 at the intersection of row number 3 and column number 6 will be selected, which is 4 as shown in figure 9.

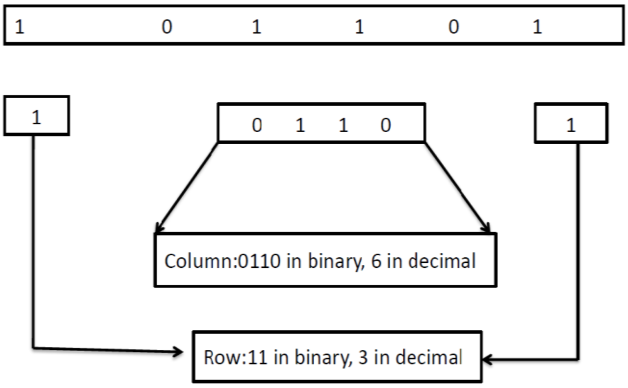
1 0 1 1 0 1
1
0 1 1 0
1
Column:0110 in binary, 6 in decimal
Row:11 in binary, 3 in decimal

Figure 9: Example of selection of S box output based on the input

The output of all the S boxes are then combined to form a 32 bit block, which is given to the next stage of a round, the P box permutation.

Step 4: P box permutation

The output of S box consists of 32 bits. These 32 bits are permuted using a P box. This permutation involves simple permutation, without any expansion or compression. This is called P box permutation. The P box permutation is shown in table 12. Here a 16 in the first block indicates that the bit at position 16 of the original input moves to bit at position 1 in the output, and a 10 in the block number 16 indicates that the bit at the position 10 of the original input moves to bit at the position 16 in the output.

Table 12: P box permutation

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Step 5: Xor and Swap

Here the left half portion (i.e. LPT) of the initial 64 bit plain text is XORed with the output produced by P box permutation. The result of this XOR operation becomes the new right half (i.e RPT). The old right half (i.e RPT) becomes the new left half, in a process of swapping. This is shown in figure 10 and can be represented as

$L_n = R_{n-1}$

$R_n = L_{n-1} \oplus f(R_{n-1}, K_n)$

Where

L_n represents the current left plain text

R_n represents the current right plain text

L_{n-1} represents the previous left plain text

R_{n-1} represents the previous right plain text

K_n represents the current key

f represents the cipher function

$\oplus$ denotes the exclusive OR (XOR) operation

f(R,k) is calculated from the figure 7.

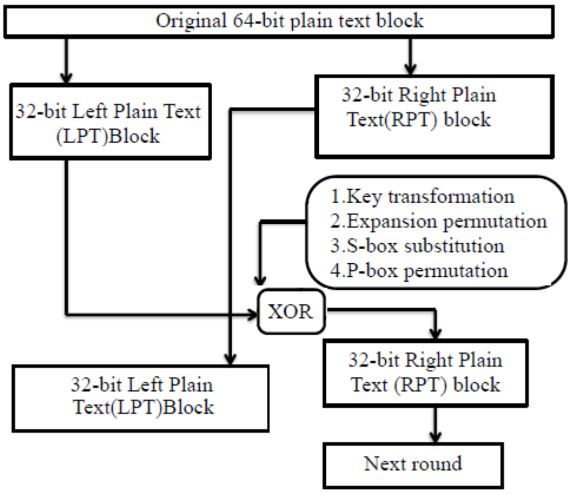
Original 64-bit plain text block
32-bit Left Plain Text (LPT)Block
32-bit Right Plain Text(RPT) block
1.Key transformation
2.Expansion permutation
3.S-box substitution
4.P-box permutation
XOR
32-bit Left Plain Text(LPT)Block
32-bit Right Plain Text (RPT) block
Next round

Figure 10: Xor and Swap

Step 6: Final permutation

At the end of the 16 rounds, the final permutation is performed. The output of the final permutation is the 64 bit encrypted block. The table 13 shows the final permutation. The 40th input bit takes the position of the 1st output bit, 8th input bit takes the position of the 2^{nd} output bit and so on.

Table 13: Final permutation

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

3.3 DES decryption

The only difference between the encryption and the decryption process is the reversal of key portions. If the original key K was divided by into K1, K2, K3,....K16 for the

16 encryption rounds, then for decryption, the key should be used as K16, K15, K14,...K1.

3.4 Data compression and decompression using Run length encoding (RLE)

Run Length Encoding (RLE) is created especially for data with strings of repeated symbols. For example, using RLE the string AAABBAAACCCCBBB can be compressed into ”3A2B3A4C3B”. The string of 15 bytes can be expressed to a string of 10 bytes. The main advantage of RLE is that it performs lossless data compression in which the original data can be perfectly reconstructed from the compressed data. Run length encoding has a weakness of its reliability on the nature of the input data. In some cases, it can even give a larger compressed string than the original. For example if the input string is “ABDBAC”, the compressed string would be 1A1B1D1B1A1C, which is twice as large as the original.

In order to overcome this limitation a new data compression using RLE is used in this method. Using the data compression, the repeated four 1’s in the original message will be compressed to 0’s, such that there will be a power reduction also. Because when the 1’s are transmitted there will be high power. This can be minimized when all the 1’s are compressed to 0’s. For example let the original data be 1111111100001100. At first the input data will be divided in to 4 blocks each having 4 bits and then the data compression is applied. The first block (1111) consists of four repeated 1’s and will be compressed to four 0’s. Then the second block (1111) consists of again four repeated 1’s and compressed to four 0’s. The third block (0000) consists of four repeated 0’s and this same data will be transmitted. The fourth block (1100) consist of neither repeated 1’s nor repeated 0’s and will be transmitted as same. Finally the compressed data will be 0000000000001100.

4. Results and Discussion

The modules are modeled using VHDL in Xilinx ISE Design Suite 13.2 and the simulation of the design is performed using ModelSim SE 6.3f to verify the functionality of the design. Here a method of secure data transmission using cryptography is developed. It contains modules such as an encryption, compression, decompression, and decryption. Simulation results of these modules are shown below.

4.1 Simulation of data encryption using TDES

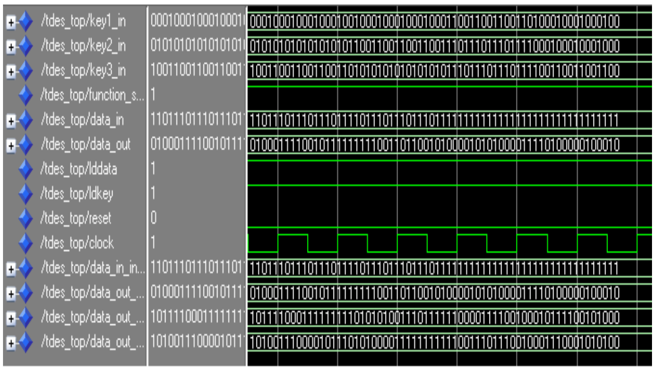
/tdes_top/key1_in 0001000100010001 0001000100010001001000100010001000110011001100110100010001000100
/tdes_top/key2_in 0101010101010101 0101010101010101011001100110011001110111011101111000100010001000
/tdes_top/key3_in 1001100110011001 1001100110011001101010101010101010111011101110111100110011001100
/tdes_top/function_s... 1
/tdes_top/data_in 1101110111011101 1101110111011101110111011101110111111111111111111111111111111111
/tdes_top/data_out 0100011110010111 0100011110010111111111001101100101000010101000011110100000100010
/tdes_top/lddata 1
/tdes_top/ldkey 1
/tdes_top/reset 0
/tdes_top/clock 1
/tdes_top/data_in_in... 1101110111011101 1101110111011101110111011101110111111111111111111111111111111111
/tdes_top/data_out_... 0100011110010111 0100011110010111111111001101100101000010101000011110100000100010
/tdes_top/data_out_... 1011110001111111 1011110001111111110101010011101111110000111100100010111100101000
/tdes_top/data_out_... 1010011100001011 1010011100001011101010000111111111100111011100100011100001010100

Figure 11: Data encryption using TDES

The 64 bit data, key 1, key 2, key 3 are the main inputs here. The function select input will be 1 for performing encryption. The output will be 64 bit encrypted data.

4.2 Simulation of data compression using RLE

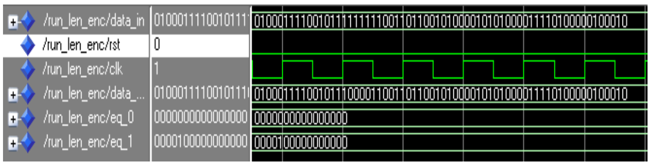
/run_len_enc/data_in 0100011110010111 0100011110010111111111001101100101000010101000011110100000100010
/run_len_enc/rst 0
/run_len_enc/clk 1
/run_len_enc/data_... 0100011110010111 0100011110010111000011001101100101000010101000011110100000100010
/run_len_enc/eq_0 0000000000000000 0000000000000000
/run_len_enc/eq_1 0000100000000000 0000100000000000

Figure 12: Data compression using RLE

The encrypted data will be the input for data compression and the output will be the compressed data. The eq0,eq1 denotes the position where compression occurs.

4.3 Simulation of data decompression using RLE

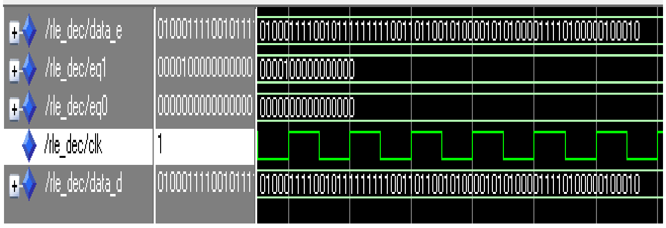
/rle_dec/data_e 0100011110010111 0100011110010111111111001101100101000010101000011110100000100010
/rle_dec/eq1 0000100000000000 0000100000000000
/rle_dec/eq0 0000000000000000 0000000000000000
/rle_dec/clk 1
/rle_dec/data_d 0100011110010111 0100011110010111111111001101100101000010101000011110100000100010

Figure 13: Data decompression using RLE

The compressed data, eq0, and eq1 are the inputs here. The output will be the decompressed data which is similar to the encrypted data.

4.4 Simulation of data decryption using TDES

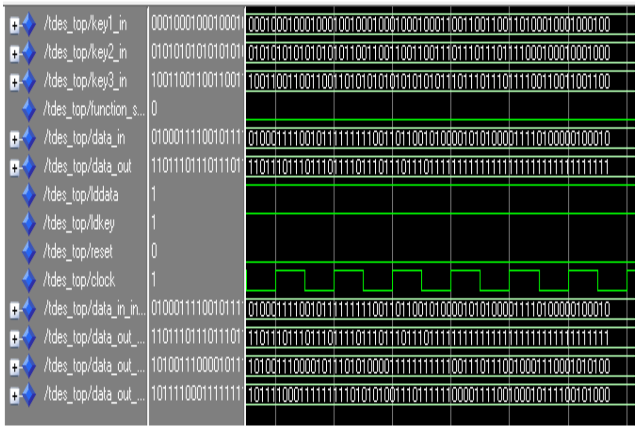
/tdes_top/key1_in 0001000100010001 0001000100010001001000100010001000110011001100110100010001000100
/tdes_top/key2_in 0101010101010101 0101010101010101011001100110011001110111011101111000100010001000
/tdes_top/key3_in 1001100110011001 1001100110011001101010101010101010111011101110111100110011001100
/tdes_top/function_s... 0
/tdes_top/data_in 0100011110010111 0100011110010111111111001101100101000010101000011110100000100010
/tdes_top/data_out 1101110111011101 1101110111011101110111011101110111111111111111111111111111111111
/tdes_top/lddata 1
/tdes_top/ldkey 1
/tdes_top/reset 0
/tdes_top/clock 1
/tdes_top/data_in_in... 0100011110010111 0100011110010111111111001101100101000010101000011110100000100010
/tdes_top/data_out_... 1101110111011101 1101110111011101110111011101110111111111111111111111111111111111
/tdes_top/data_out_... 1010011100001011 1010011100001011101010000111111111100111011100100011100001010100
/tdes_top/data_out_... 1011110001111111 1011110001111111110101010011101111110000111100100010111100101000

Figure 14: Data decryption using TDES

The decompressed data, key1, key2, key3 will be the main inputs for the decryption process. The function select input will be 0 for performing decryption. The output will be the 64 bit original input.

5. Conclusion

The use of network and communication facilities are increasing day by day therefore secure data transmission is important in modern life. This work intends to develop a secure data transmission facility over an insecure channel. The cryptography plays an important role in secure data transmission so that data can be encrypted with the key which can be decrypted only by the intended receiver. In the traditional method of secure data transmission, the data is first compressed, encrypted and then send to the receiver. But this method is not convenient if the sender is not able to perform compression first as it requires additional hardware. So in order to overcome this problem, the existing method of data transmission is reversed such that the data will be encrypted first and then compressed without compromising the information secrecy. In this method, if the sender is not able to perform compression first then he or she can use the help of a third party because when the data is first encrypted with a secret key then the data become secure so that an intruder or a third person cannot read the original message. Here the original data is encrypted first using triple DES algorithm. The key size of triple DES algorithm is large, so that an intruder cannot easily attack the original data. After encryption, the encrypted data is compressed using RLE and then transmitted to the receiver. At the receiving section, the data is first decompressed using RLE and then decrypted using triple DES algorithm to retrieve the original message. This work is synthesized in Xilinx ISE design suite 13.2 and simulated in ModelSim 6.3f**.**

References

[1] Demijan Klinc, Carmit Hazay, Ashish Jagmohan, Hugo Krawczyk, Tal Rabin, "On compression of data encrypted with block ciphers", IEEE Transactions On Information Theory, Vol. 58, No. 11, November 2012.

[2] Chethan Kumar K V, S Sujatha, "VLSI Implementation of DES and TDES Algorithm with Cipher Block Concept" International Journal of Emerging Science and Engineering (IJESE) Volume 2, Issue 7, May 2014.

[3] Amandeep Singh, Manu Bansal, "FPGA implementation of optimized DES encryption algorithm on spartan 3E", International journal of scientific and engineering research, vol. 1, issue 1, Oct. 2013.

[4] Sombir Singh, Sunil K Maakar, Dr. Sudhesh Kumar Gupta, "Enhancing the Security of DES Algorithm Using Transposition Cryptography Techniques" International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, issue 6, 2013.

[5] Aqib Al Azad, "Efficient VLSI Implementation of DES and Triple DES Algorithm with Cipher Block Chaining concept using Verilog and FPGA" International Journal of Computer Applications volume 44, no 16, April 2012.

[6] Sai Praveen Venigalla, M. Nagesh Babu, Srinivas Boddu, G. Santhi Swaroop Vemana, "Implementation of the Triple DES block cipher using VHDL", International Journal of Advances in Engineering and Technology, vol. 3, issue 1, March 2012.

[7] Jawahar Thakur, Nagesh Kumar, "DES, AES : Symmetric key cryptography algorithms simulation based performance analysis", International journal of emerging technology and advanced engineering, vo1. l, issue 2, december 2011.

[8] P. Kitsos, S Goudevenos, O. Koufopavlou, "VLSI Implementations of the Triple DES block cipher", IEEE Transactions on VLSI, Vol.45, No.9, April 2003.

[9] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption : Analysis of the DES modes of operation", presented at the IEEE 38th Annu. Symp. Found. Comput. Sci., 1997.

[10] Nimmi Gupta, "Implementation of optimized DES encryption algorithm upto 4 round on spartan 3", International journal of computer technology and electronics engineering (IJCTEE), vol. 2, issue 1.

[11] Simon Haykin, "Communication System", 4th edition, Hamilt on printing company.