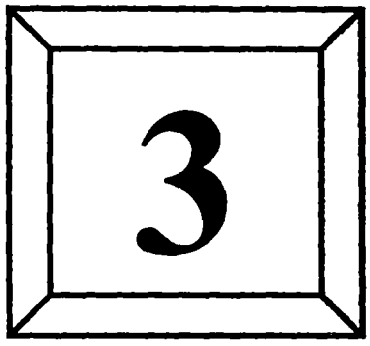


Encryption Through Triangular Encryption (TE) Technique

	Contents	Pages

3.1 Introduction

In chapter 2, the RPSP technique has been discussed in detail. In this chapter, the Triangular Encryption (TE) Technique is proposed.

The nomenclature followed for this proposed technique, Triangular Encryption (TE) Technique, is on the basis of the structure that is supposed to be formed out of the source as well as different intermediate and final blocks of bits during the process of encryption. In fact, an equilateral triangular shape is formed if the source block of bits and different intermediate blocks along with the final 1-bit block are shown in line-by-line manner.

Now, the basic characteristic of this TE technique, in which it is entirely different from the RPSP technique, discussed in chapter 2, is the use of the Boolean operation. Also, unlike the RPSP technique, here no attempt is made for the formation of a cycle. Another important aspect of this TE technique is that here there is no question of the positional reorientation of bits; rather all the bits in a block are directly participating in a Boolean operation.

Since, like all the other proposed techniques, this TE technique is also a bit-level technique, the stream of bits corresponding to the source file to be encrypted is to be decomposed into a finite number of blocks that are advised to be of varying lengths. For each of the blocks, the technique of TE is to be applied to generate the corresponding target block.

Now, in the Triangular Encryption (TE) technique, from a source block of size, say, n, an intermediate block of size (n-1) is generated by applying the exclusive OR (XOR) operation between each two consecutive bits. The same process is again applied to the generated block of size (N-1) to generate a block of size (n-2). The process goes on until the generation of a 1-bit block. All these blocks under consideration together form an equilateral triangle-like shape.

After the formation of such a triangular shape, putting together either the MSBs or the LSBs of all the blocks under consideration in either sequence, the target block is formed. In this regard, the key takes a vital role because only by knowing this key the receiver of the message can understand on how the target block is chosen from the triangular shape. Obviously, this key is to be kept secret. So it is a secret key system.

The encrypted stream of bits is generated by putting together all the target blocks [2, 48, 57].

Section 3.2 of this chapter discusses the scheme followed in this technique through algorithmic presentation as well as diagrammatic representation. An implementation of this technique is described in section 3.3. Section 3.4 shows results after implementing this technique in a certain manner on the same set of files that was also considered for the RPSP technique in chapter 2. An analytical view and the concluding remark on this technique are drawn in section 3.5.

3.2 The Scheme

The plaintext to be transmitted is to be converted into a stream of bits. Since the TE technique, like all the other proposed techniques, is a bit-level technique, here also the source stream is to be decomposed into a finite number of blocks, not necessarily of the same length. In fact, as it will be analyzed in section 3.5, an enhancement of security can be done by allowing block sizes to be different because it makes the key length large enough, hereby almost nullifying the chance of cryptanalysis to break the cipher [48].

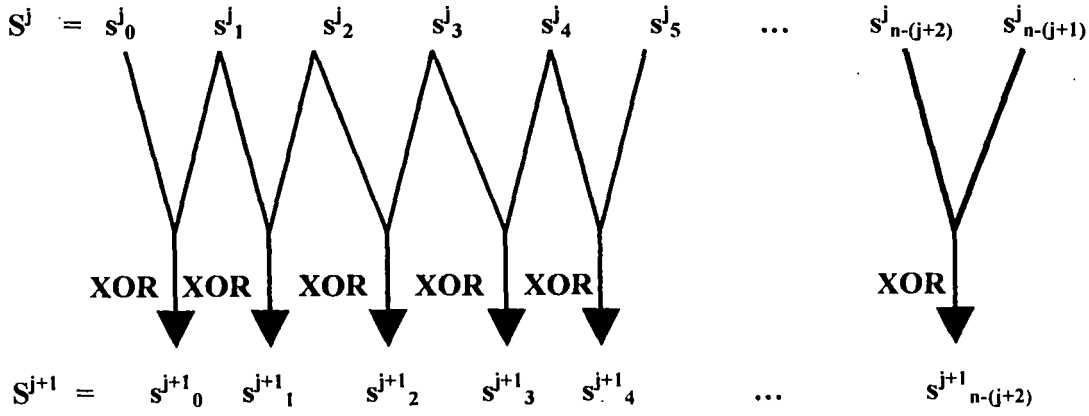
The entire scheme includes several parts in it. First of all is the formation of the triangle, which is shown in section 3.2.1. Section 3.2.2 discusses different options that are available to form the target block from the triangle generated in section 3.2.1. Section 3.2.3 gives the processes for decryption to generate the source block from a target block. Section 3.2.4 gives a simple example to illustrate the scheme.

3.2.1 Formation of Triangle

Consider a block $S = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4\ s^0_5 \ldots s^0_{n-2}\ s^0_{n-1}$ of size n bits, where $s^0_i = 0$ or 1 for $0 <= i <= (n-1)$.

Starting from MSB (s^0_0) and the next-to-MSB (s^0_1), bits are pair-wise XORed, so that the 1st intermediate sub-stream $S^1 = s^1_0\ s^1_1\ s^1_2\ s^1_3\ s^1_4\ s^1_5 \ldots s^1_{n-2}$ is generated consisting of (n-1) bits, where $s^1_j = s^0_j \oplus s^0_{j+1}$ for $0 <= j <= n-2$, $\oplus$ stands for the exclusive OR operation. This 1st intermediate sub-stream S^1 is also then pair-wise XORed to generate $S^2 = s^2_0\ s^2_1\ s^2_2\ s^2_3\ s^2_4\ s^2_5 \ldots s^2_{n-3}$, which is the 2nd intermediate sub-stream of length (n-2). This process continues (n-1) times to ultimately generate $S^{n-1} = s^{n-1}_0$,

which is a single bit only. Thus the size of the 1st intermediate sub-stream is one bit less than the source sub-stream; the size of each of the intermediate sub-streams starting from the 2nd one is one bit less than that of the sub-stream wherefrom it was generated; and finally the size of the final sub-stream in the process is one bit less than the final intermediate sub-stream. Figure 3.2.1.1 shows the generation of an intermediate sub-stream $S^{j+1} = s^{j+1}_0\ s^{j+1}_1\ s^{j+1}_2\ s^{j+1}_3\ s^{j+1}_4\ s^{j+1}_5 \ldots s^{j+1}_{n-(j+2)}$ from the previous intermediate sub-stream $S^j = s^j_0\ s^j_1\ s^j_2\ s^j_3\ s^j_4\ s^j_5 \ldots s^j_{n-(j+1)}$. The formation of the triangular shape for the source sub-stream $S = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4\ s^0_5 \ldots s^0_{n-2}\ s^0_{n-1}$ is shown in figure 3.2.1.2.



Generation of an Intermediate Sub-Stream in TE
Figure 3.2.1.1

$S = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4\ s^0_5 \ldots s^0_{n-2}\ s^0_{n-1}$

$S^1 = s^1_0\ s^1_1\ s^1_2\ s^1_3\ s^1_4 \ldots s^1_{n-2}$

$S^2 = s^2_0\ s^2_1\ s^2_2\ s^2_3 \ldots s^2_{n-3}$

$S^3 = s^3_0\ s^3_1\ s^3_2 \ldots s^3_{n-4}$

$S^4 = s^4_0\ s^4_1 \ldots s^4_{n-5}$

$S^5 = s^5_0 \ldots s^5_{n-6}$

$\ldots\ \ldots\ \ldots\ \ldots\ \ldots$

$S^{n-2} = s^{n-2}_0\ s^{n-2}_1$

$S^{n-1} = s^{n-1}_0$

Formation of a Triangle in TE
Figure 3.2.1.2

3.2.2 Options for Forming Target Blocks from Triangle

Corresponding to figure 3.2.1.2, a total of four options are available to form a target block from the source block $S = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4\ s^0_5 \ldots\ s^0_{n-2}\ s^0_{n-1}$. These options are shown in table 3.2.2.1 [48]. Figure 3.2.2.1 shows the same diagrammatically.

Table 3.2.2.1
Options for choosing Target Block from Triangle

Option Serial No.	Target Block	Method of Formation
001	$s^0_0\ s^1_0\ s^2_0\ s^3_0\ s^4_0\ s^5_0 \ldots\ s^{n-2}_0\ s^{n-1}_0$	Taking all the MSBs starting from the source block till the last block generated
010	$s^{n-1}_0\ s^{n-2}_0\ s^{n-3}_0\ s^{n-4}_0\ s^{n-5}_0 \ldots\ s^1_0\ s^0_0$	Taking all the MSBs starting from the last block generated till the source block
011	$s^0_{n-1}\ s^1_{n-2}\ s^2_{n-3}\ s^3_{n-4}\ s^4_{n-5} \ldots\ s^{n-2}_1\ s^{n-1}_0$	Taking all the LSBs starting from the source block till the last block generated
100	$s^{n-1}_0\ s^{n-2}_1\ s^{n-3}_2\ s^{n-4}_3\ s^{n-5}_4 \ldots\ s^1_{n-2}\ s^0_{n-1}$	Taking all the LSBs starting from the last block generated till the source block

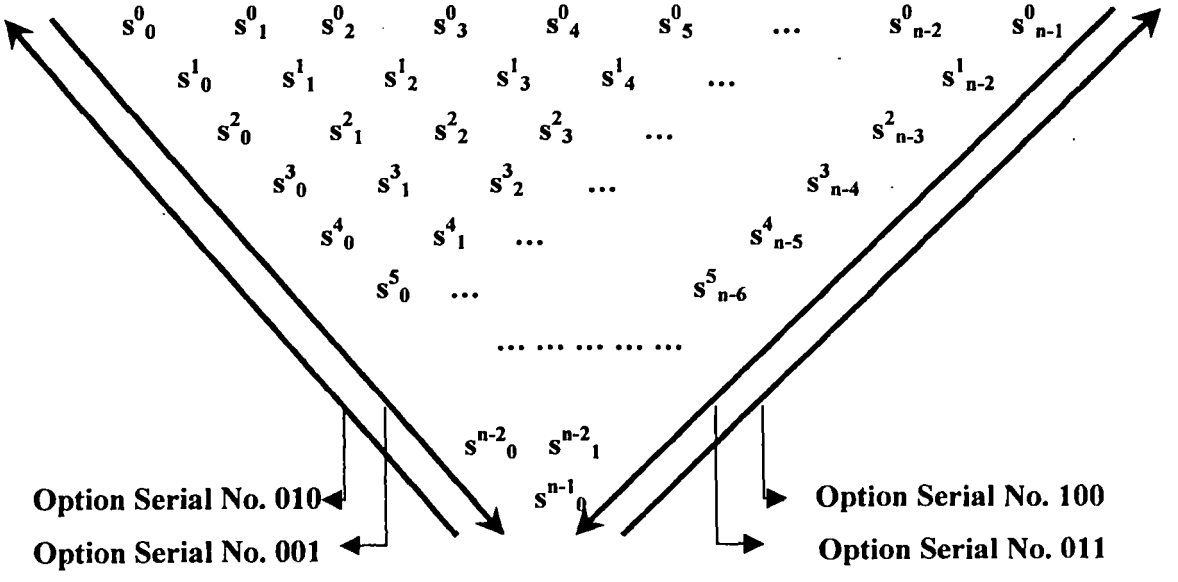


Figure 3.2.2.1
Diagrammatic Representation of Options for choosing Target Block from Triangle

3.2.3 Generating Source Block from a Target Block

For the purpose of generating the source block from a target block the reference to table 3.2.3.1 is important.

Table 3.2.3.1
Decryption Reference for Different Target Blocks

Corresponding Option Serial No.	Decryption Reference	Comment
001	Symmetric	Exactly the same encryption process to be followed
010	Asymmetric	To follow a different technique for decryption
011	Asymmetric	To follow a different technique for decryption
100	Symmetric	Exactly the same encryption process to be followed

As is shown in table 3.2.3.1, corresponding to option 001 and option 100, the processes of generating the source block are the same, which means, forming the triangle and picking bits in the same manner one by one.

Section 3.2.3.1 discusses how the source block is generated from the target block corresponding to the option serial no. 010 and the same for the target block corresponding to the option serial no. 011 is discussed in section 3.2.3.2.

3.2.3.1 Generating Source Block from Target Block $s^{n-1}_0\ s^{n-2}_0\ s^{n-3}_0\ s^{n-4}_0\ s^{n-5}_0 \ldots s^1_0\ s^0_0$ (With Option Serial No. 010)

As shown in table 3.2.2.1, the encrypted sub-stream corresponding to the option serial no. 010 is $s^{n-1}_0\ s^{n-2}_0\ s^{n-3}_0\ s^{n-4}_0\ s^{n-5}_0 \ldots s^1_0\ s^0_0$.

To ease the explanation of decryption technique, let us consider, $e^0_{i-1} = s^{n-i}_0$ for $1 <= i <= n$, so that the target block corresponding to the option serial no. 2 becomes $E = e^0_0\ e^0_1\ e^0_2\ e^0_3\ e^0_4 \ldots e^0_{n-2}\ e^0_{n-1}$. Now, following the same approach as mentioned in section 3.2.1, a triangle is to be formed. After the formation of the triangle, for the purpose of decryption, the block $e^0_{n-1}\ e^1_{n-2}\ e^2_{n-3}\ e^3_{n-4}\ e^4_{n-5}\ e^5_{n-6} \ldots e^{n-2}_1\ e^{n-1}_0$, i.e., the sub-stream taking all the LSBs of the blocks starting from E to the finally generated 1-bit block E^{n-1}, are to be taken together and it is to be considered as the decrypted block. Figure 3.2.3.1.1 shows the triangle generated and hence the decrypted block obtained. Here the intermediate blocks are referred to as E^1, E^2, ..., E^{n-2} and the final block generated as E^{n-1}.

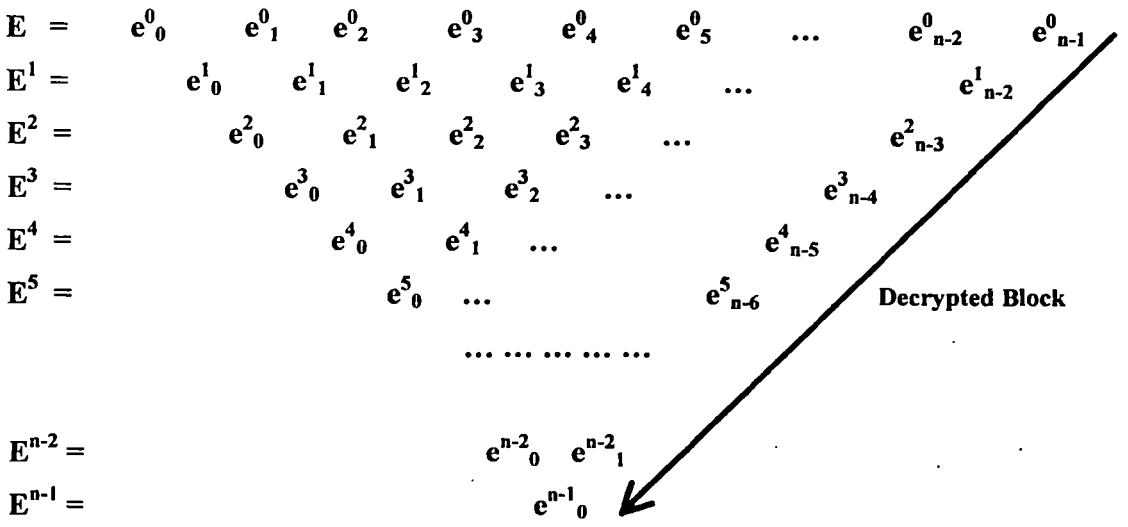


Figure 3.2.3.1.1
Generation of Source Block from Target Block with Option Serial No. 010

3.2.3.2 Generating Source Block from Target Block $s^0_{n-1}\ s^1_{n-2}\ s^2_{n-3}\ s^3_{n-4}\ s^4_{n-5}\ \ldots\ s^{n-2}_1\ s^{n-1}_0$ (With Option Serial No. 011)

As shown in table 3.2.2.1, the encrypted block corresponding to the option serial no. 011 is $s^0_{n-1}\ s^1_{n-2}\ s^2_{n-3}\ s^3_{n-4}\ s^4_{n-5}\ \ldots\ s^{n-2}_1\ s^{n-1}_0$.

To ease the explanation of decryption technique, let us consider, $e^0_{i-1} = s^{i-1}_{n-i}$ for $1 <= i <= n$, so that the encrypted block becomes $E = e^0_0\ e^0_1\ e^0_2\ e^0_3\ e^0_4 \ldots e^0_{n-2}\ e^0_{n-1}$. Now, following the same approach as mentioned in section 3.2.1, a triangle is to be formed. After the formation of the triangle, for the purpose of decryption, the block $e^{n-1}_0\ e^{n-2}_0\ e^{n-3}_0\ e^{n-4}_0\ e^{n-5}_0 \ldots e^1_0\ e^0_0$, i.e., the block constructed by taking all the MSBs of the blocks starting from the finally generated single-bit block E^{n-1} to E, are to be taken together and it is to be considered as the decrypted block. Figure 3.2.3.2.1 shows the triangle generated and hence the decrypted block obtained. Here the intermediate blocks are referred to as $E^1, E^2, \ldots, E^{n-2}$ and the final block generated as E^{n-1}.

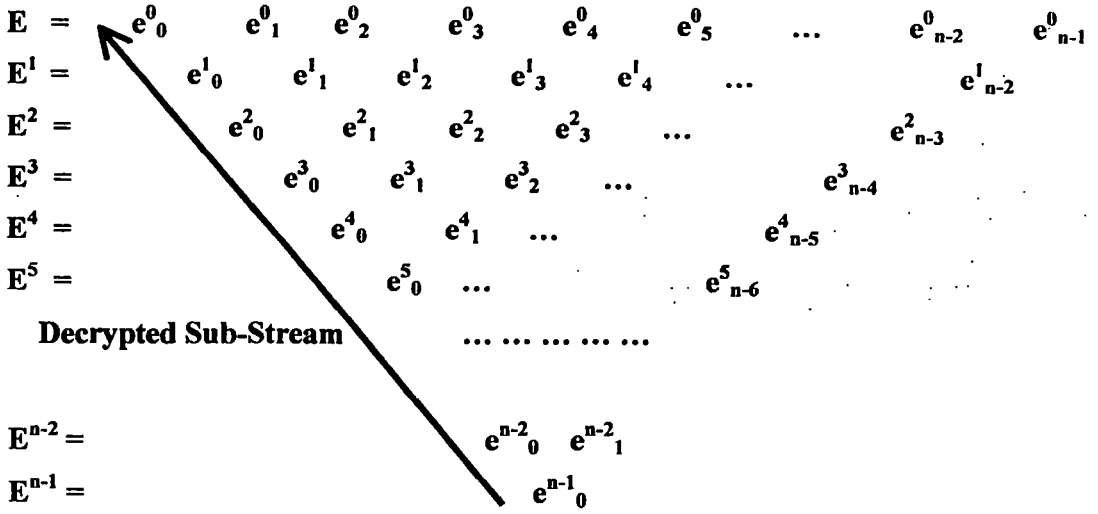


Figure 3.2.3.2.1
Generation of Source Block from Target Block with Option Serial No. 011

3.2.4 A Sample Example to Illustrate the Scheme

We consider an 8-bit block S = 10010101. Figure 3.2.4.1 shows the triangle generated using TE.

1 0 0 1 0 1 0 1
1 0 1 1 1 1 1
1 1 0 0 0 0
0 1 0 0 0
1 1 0 0
0 1 0
1 1
0

Figure 3.2.4.1
Formation of Triangle for S = 10010101

Now, from the triangle shown in figure 3.2.4.1, four types of target blocks, as are shown in table 3.2.4.1, can be generated corresponding to four options mentioned in table 3.2.2.1.

Table 3.2.4.1
Different Target Blocks generated using TE for S = 10010101

Source Block S	Target Block Corresponding to Serial No.	Target Block T
10010101	001	11101010
	010	01010111
	011	11000010
	100	01000011

For target blocks T1 = 11101010 and T_4 = 01000011, the same approach is to be followed to generate the corresponding source blocks. But blocks T_2 = 01010111 and T_3 = 11000010 require different techniques following sections 3.2.3.1 and 3.2.3.2. Figure 3.2.4.2 and figure 3.2.4.3 respectively show the generations of source blocks from target blocks T_2 and T_3.

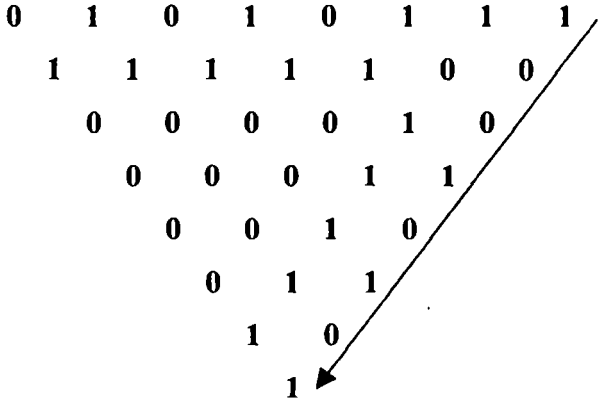


Figure 3.2.4.2
Generating Source Block S = 10010101 from Target Block T_2 = 01010111

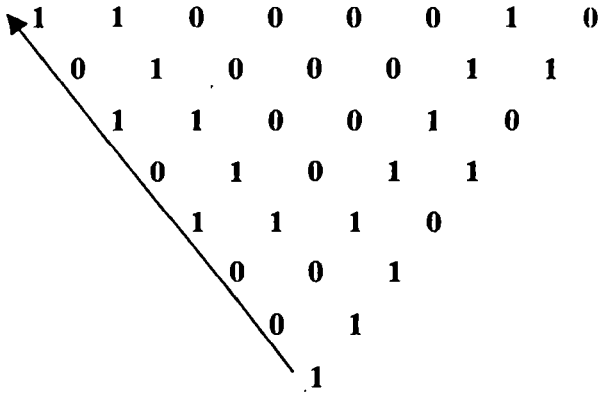


Figure 3.2.4.3
Generating Source Block S = 10010101 from Target Block T_3 = 11000010

3.3 Implementation

Consider the following confidential message to be transmitted from one source to its destination point:

"TERRORIST ATTACK SUSPECTED HOTEL SNOWVIEW DEC 06 ALERT"

To ease the implementation, say, the following are the rules applied for transmission of such a confidential message:

- The maximum number of characters allowed in the message is 60.
- The maximum length of a block is 32 bits.
- The maximum number of blocks is 20.

In section 8.2.2, in chapter 8, the proposed structure of the corresponding 180-bit key has been proposed.

Table 3.3.1 enlists the considerations to be followed during the process of encryption.

Table 3.3.1
Different Considerations for Encryption using TE

Total Number of Blocks: 16							
Size of Block 1	32	Size of Block 5	24	Size of Block 9	32	Size of Block 13	24
Option chosen	001	Option chosen	001	Option chosen	010	Option chosen	001
Size of Block 2	32	Size of Block 6	24	Size of Block 10	32	Size of Block 14	16
Option chosen	001	Option chosen	100	Option chosen	011	Option chosen	100
Size of Block 3	32	Size of Block 7	24	Size of Block 11	32	Size of Block 15	24
Option chosen	100	Option chosen	100	Option chosen	011	Option chosen	100
Size of Block 4	32	Size of Block 8	24	Size of Block 12	16	Size of Block 16	32
Option chosen	010	Option chosen	100	Option chosen	011	Option chosen	100

Following the format of the 180-bit secret key shown in figure 8.2.2.1 in chapter 8, the key will be as shown in figure 3.3.2.

The Secret Key
(Using "/" as Separators)

**100000/001/100000/001/100000/100/100000/010/011000/001/011000/100/011000/10
0/011000/100/100000/010/100000/011/100000/011/010000/011/011000/001/010000/
100/011000/100/100000/100/000000/000/000000/000/000000/000/000000/000**

The Secret Key
(Removing "/")

**100000001100000001100000100100000010011000001011000100011000100011000
100100000010100000011100000011010000011011000001010000100011000100100
000100000000000000000000000000000000000000**

Figure 3.3.2
180-bit Secret Key

Table 3.3.2 converts the message to be transmitted into the corresponding stream of bits. In this table, different characters of the message are given in column wise manner.

Table 3.3.2
Converting Characters into Bytes

Character	Byte	Character	Byte	Character	Byte	Character	Byte
T	01010100	K	01001011	E	01000101	<Blank>	00100000
E	01000101	<Blank>	00100000	L	01001100	0	00110000
R	01010010	S	01010011	<Blank>	00100000	6	00110110
R	01010010	U	01010101	S	01010011	<Blank>	00100000
O	01001111	S	01010011	N	01001110	A	01000001
R	01010010	P	01010000	O	01001111	L	01001100
I	01001001	E	01000101	W	01010111	E	01000101
S	01010011	C	01000011	V	01010110	R	01010010
T	01010100	T	01010100	I	01001001	T	01010100
<Blank>	00100000	E	01000101	E	01000101		
A	01000001	D	01000100	W	01010111		
T	01010100	<Blank>	00100000	<Blank>	00100000		
T	01010100	H	01001000	D	01000100		
A	01000001	O	00101111	E	01000101		
C	01000011	T	01010100	C	01000011		

Combining together all the bytes obtained from table 3.3.2, we get the following stream of bits of the length of 432 bits.

01010100/01000101/01010010/01010010/01001111/01010010/01001001/01010011/010
10011/00100000/01000001/01010100/01010100/01000001/01000011/01001011/001000
00/01010011/01010101/01010011/01010000/01000101/01000011/01010100/01000101/
01000100/00100000/01001000/00101111/01010100/01000101/01001100/00100000/010
10011/01001110/01001111/01010111/01010110/01001001/01000101/01010111/001000
00/01000100/01000101/01000011/00100000/00110000/00110110/00100000/01000001/
01001100/01000101/01010010/01010100

"/" being working as the separator between two consecutive bytes.

Combining the information of table 3.3.1 and table 3.3.2, we obtain table 3.3.3, in which all the blocks to be considered are mentioned.

Table 3.3.3
Different Blocks Considered for Encryption

Block 1	01010100/01000101/01010010/01010010	**Block 9**	00101111/01010100/01000101/01001100
Option	001	**Option**	010
Block 2	01001111/01010010/01001001/01010011	**Block 10**	00100000/01010011/01001110/01001111
Option	001	**Option**	011
Block 3	01010011/00100000/01000001/01010100	**Block 11**	01010111/01010110/01001001/01000101
Option	100	**Option**	011
Block 4	01010100/01000 1/01000011/01001011	**Block 12**	01010111/00100000
Option	010	**Option**	011
Block 5	00100000/01010011/01010101	**Block 13**	01000100/01000101/01000011
Option	001	**Option**	001
Block 6	01010011/01010000/01000101	**Block 14**	00100000/00110000
Option	100	**Option**	100
Block 7	01000011/01010100/01000101	**Block 15**	00110110/00100000/01000001
Option	100	**Option**	100
Block 8	01000100/00100000/01001000	**Block 16**	01001100/01000101/01010010/01010100
Option	100	**Option**	100

Figure 3.3.3 to figure 3.3.18 show the formations of all the triangles corresponding to block 1 to block 16 respectively. Along with the triangle, each figure also shows the corresponding target block following the option mentioned against each block in table 3.3.3.

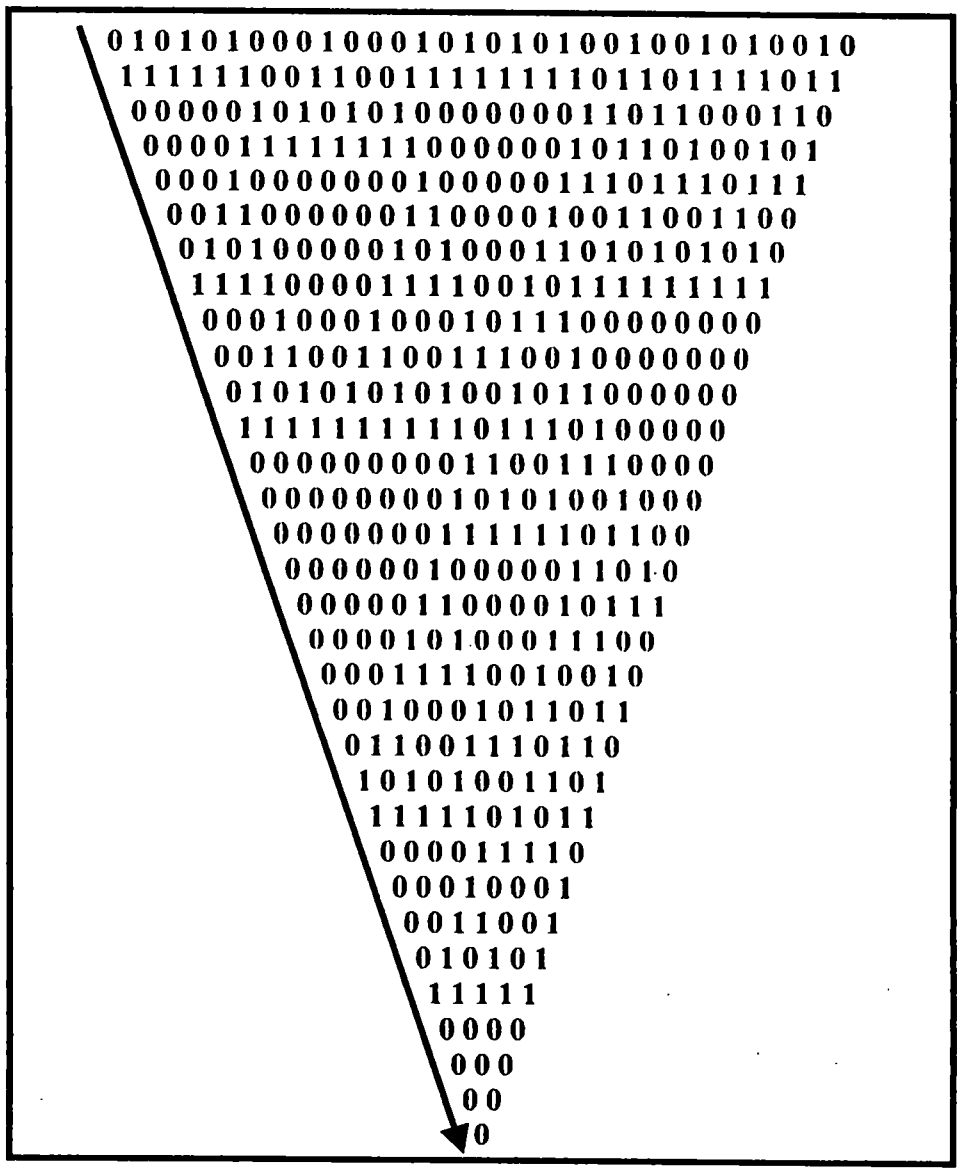


Figure 3.3.3
Construction of Target Block for Block 1 from the Generated Triangle

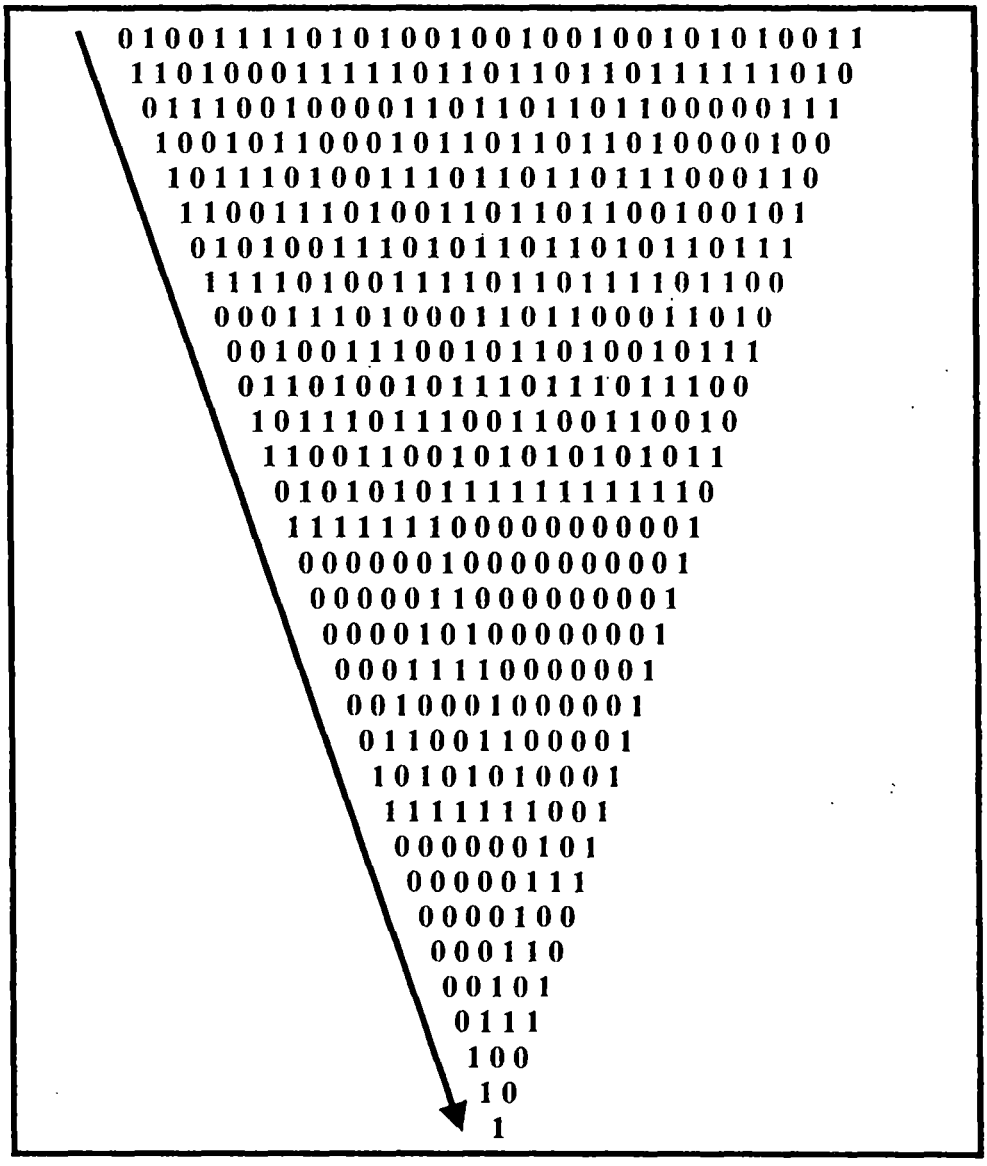


Figure 3.3.4
Construction of Target Block for Block 2 from the Generated Triangle

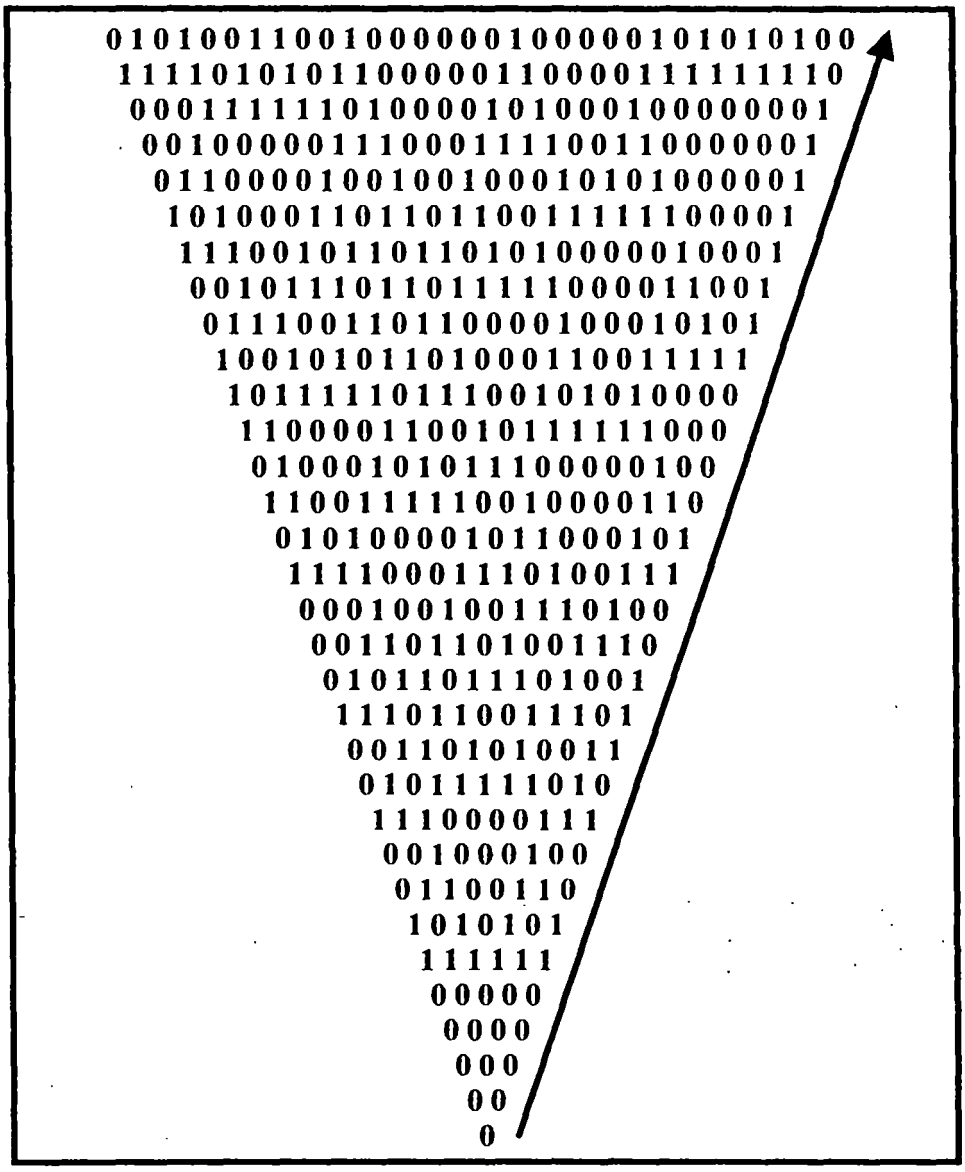


Figure 3.3.5
Construction of Target Block for Block 3 from the Generated Triangle

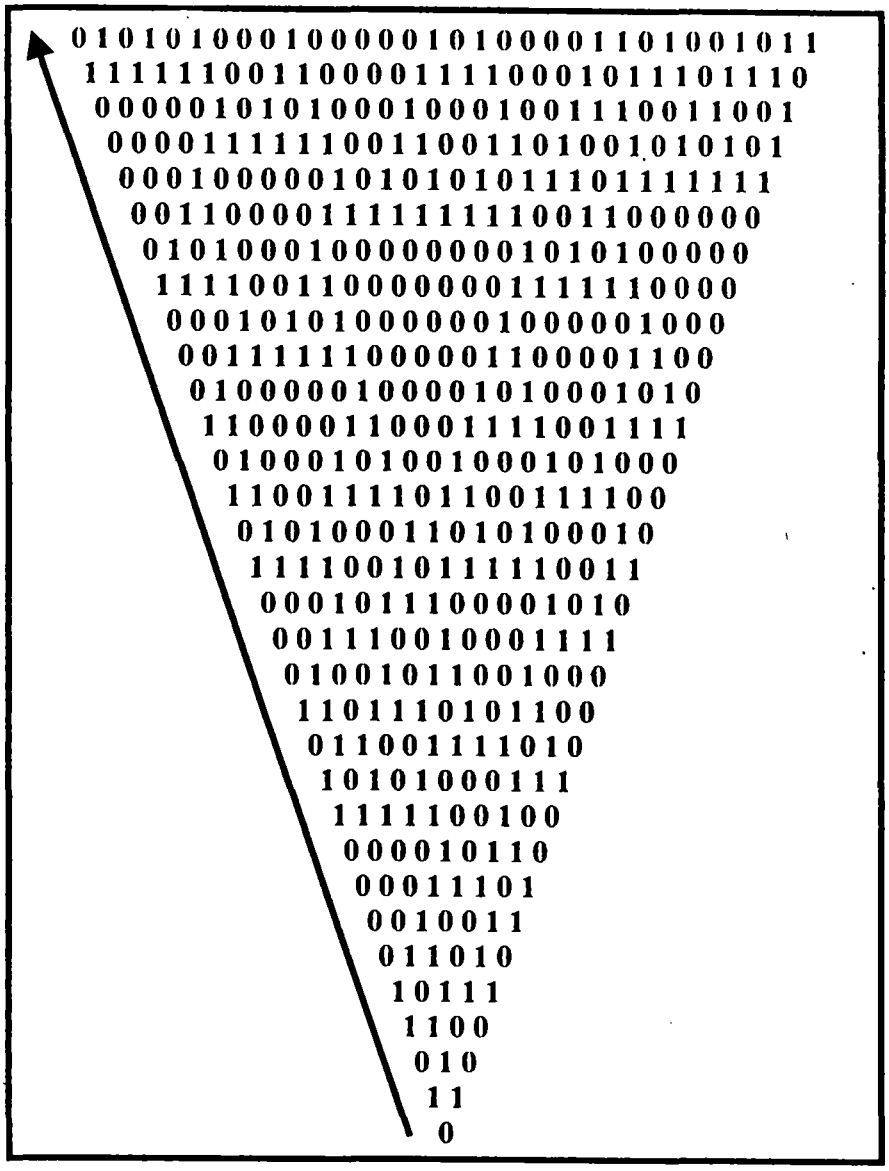


Figure 3.3.6
Construction of Target Block for Block 4 from the Generated Triangle

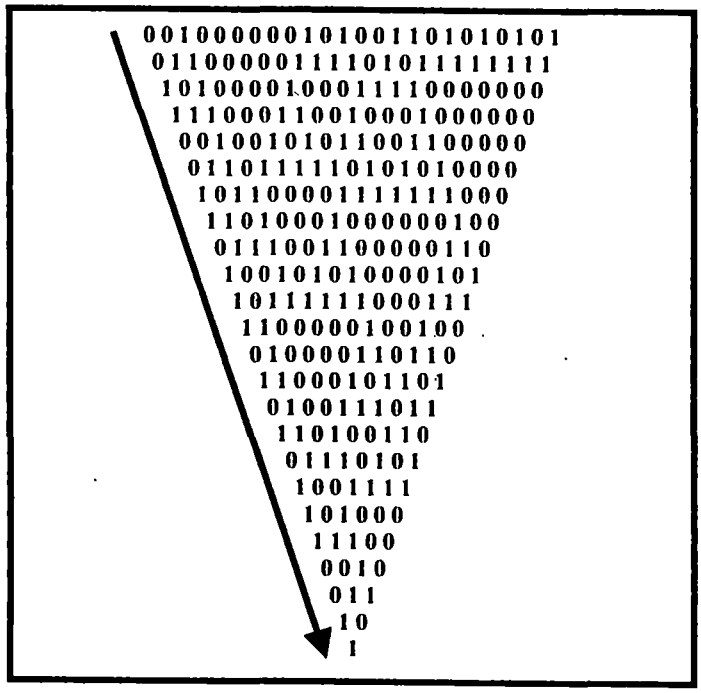


Figure 3.3.7
Construction of Target Block for Block 5 from the Generated Triangle

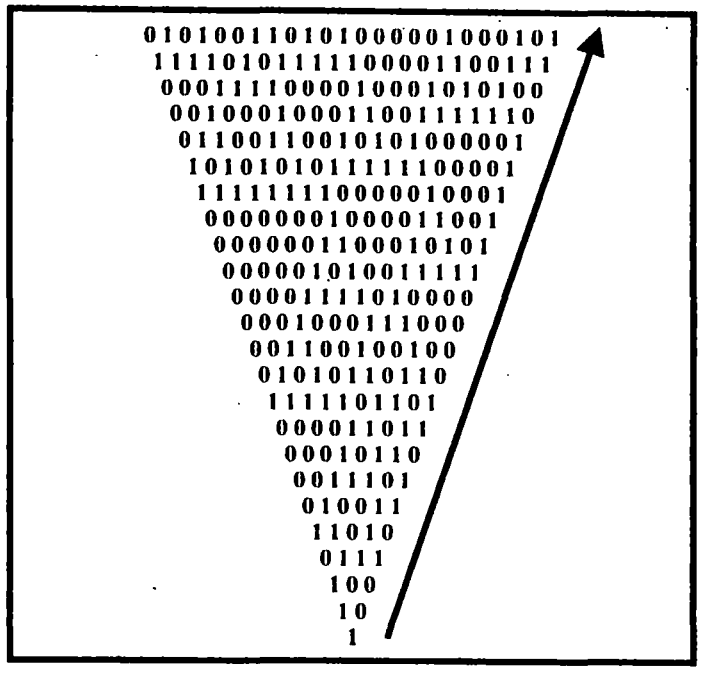


Figure 3.3.8
Construction of Target Block for Block 6 from the Generated Triangle

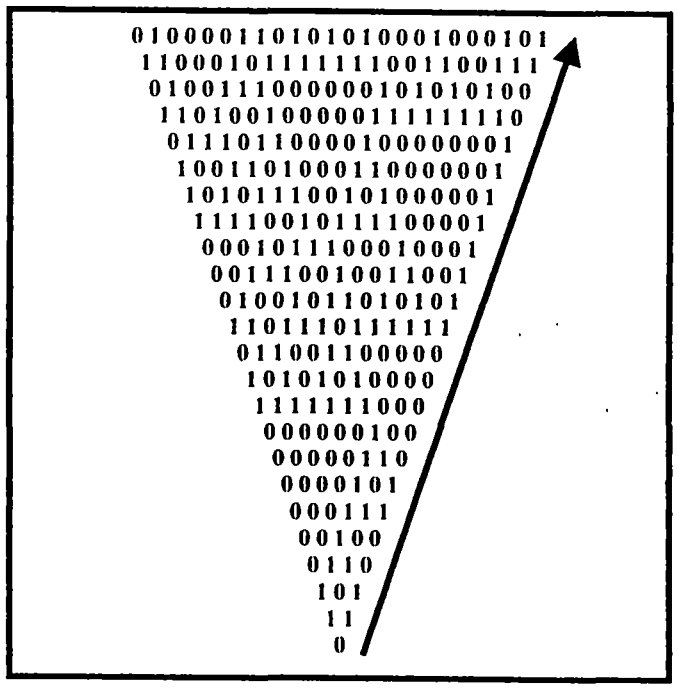


Figure 3.3.9
Construction of Target Block for Block 7 from the Generated Triangle

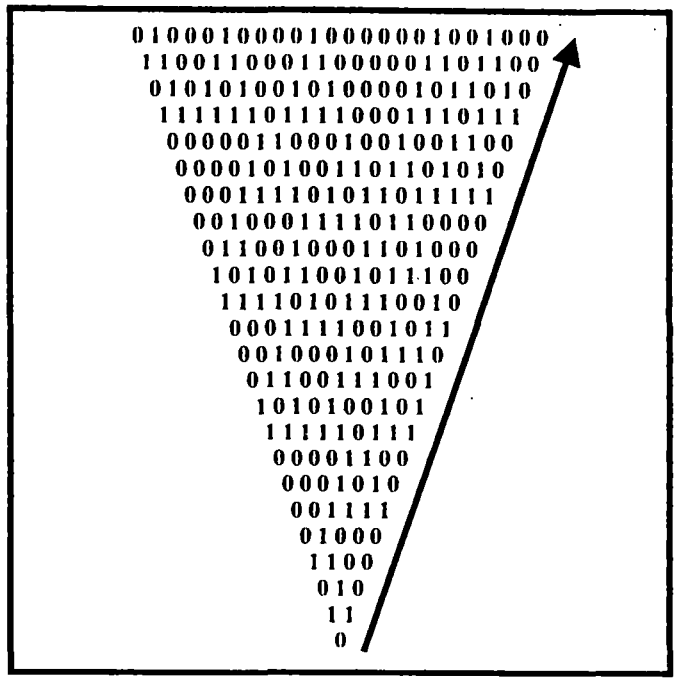


Figure 3.3.10
Construction of Target Block for Block 8 from the Generated Triangle

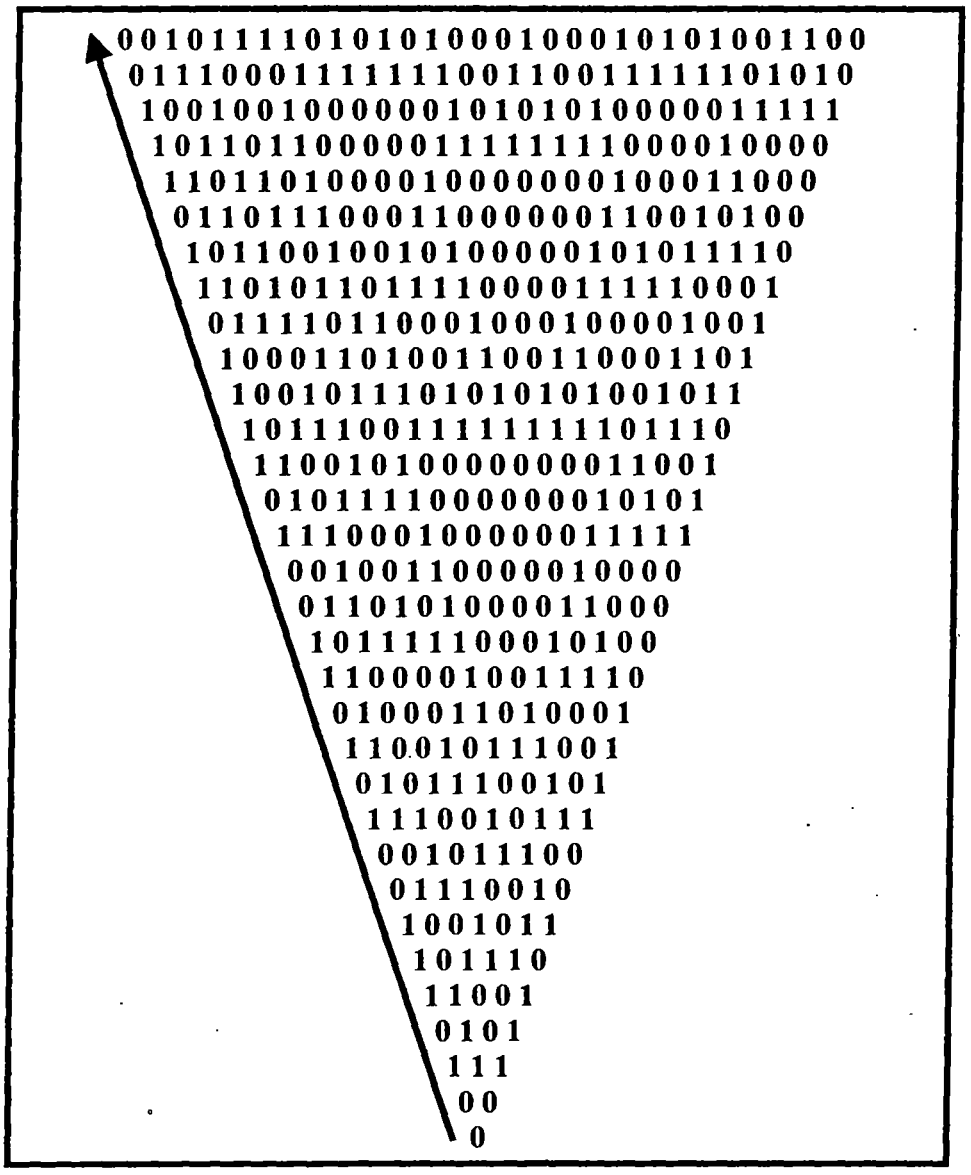


Figure 3.3.11
Construction of Target Block for Block 9 from the Generated Triangle

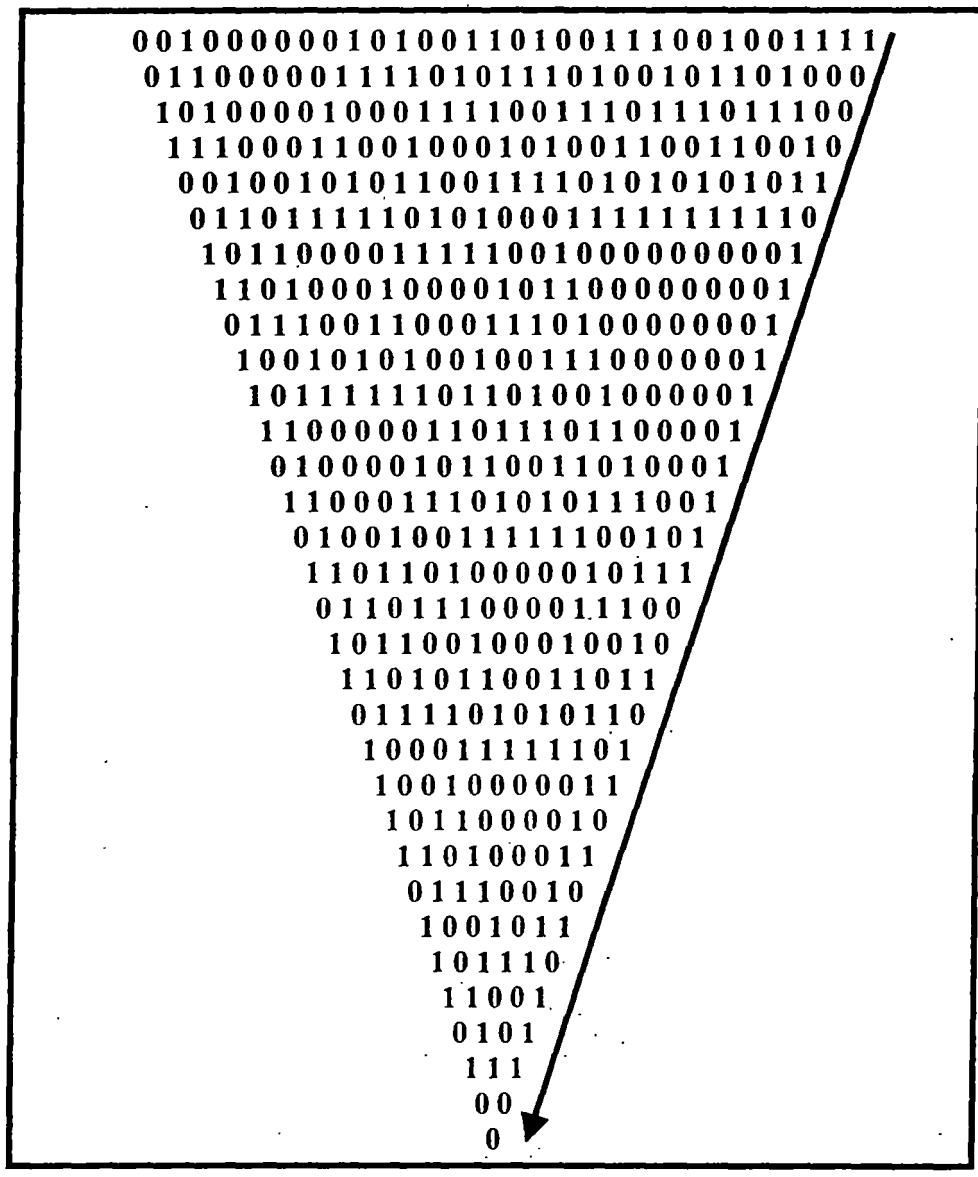


Figure 3.3.12
Construction of Target Block for Block 10 from the Generated Triangle

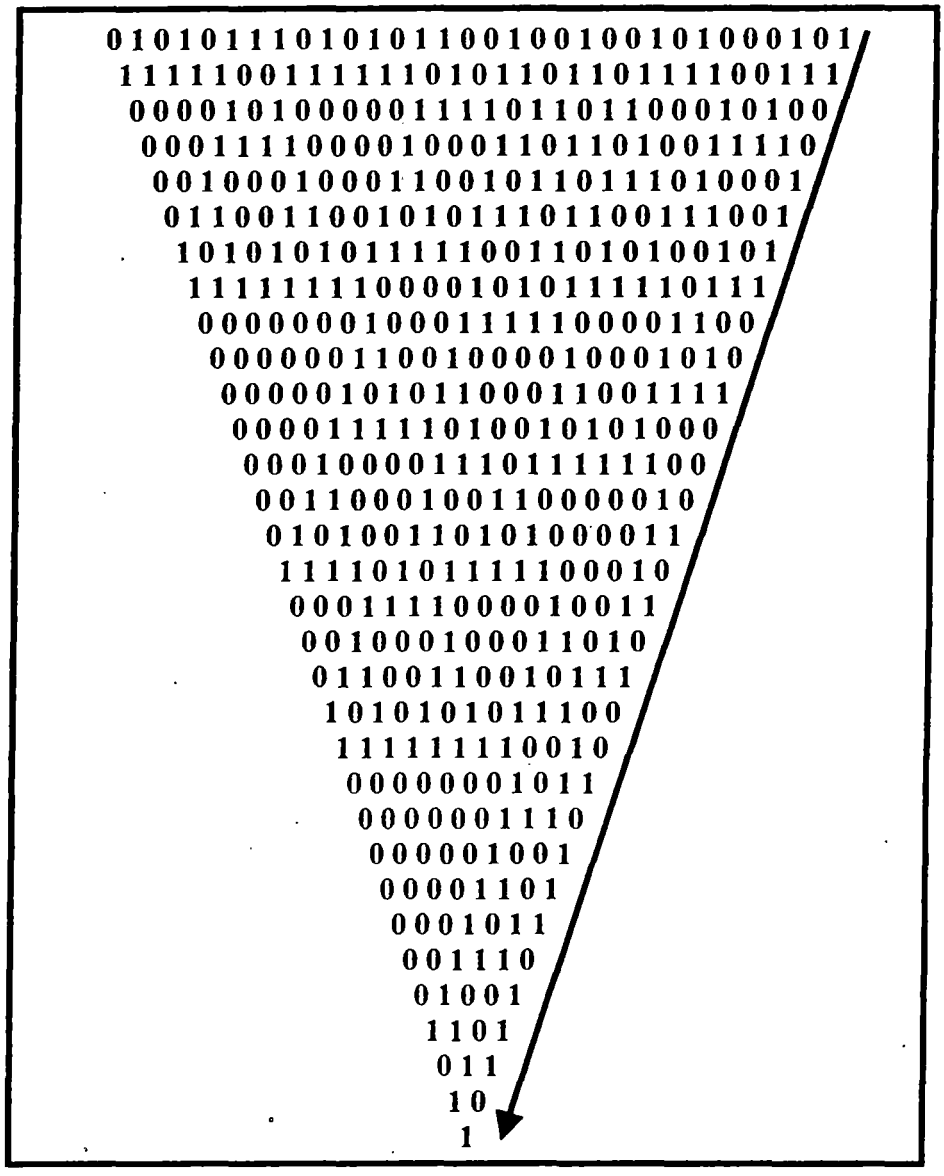


Figure 3.3.13
Construction of Target Block for Block 11 from the Generated Triangle

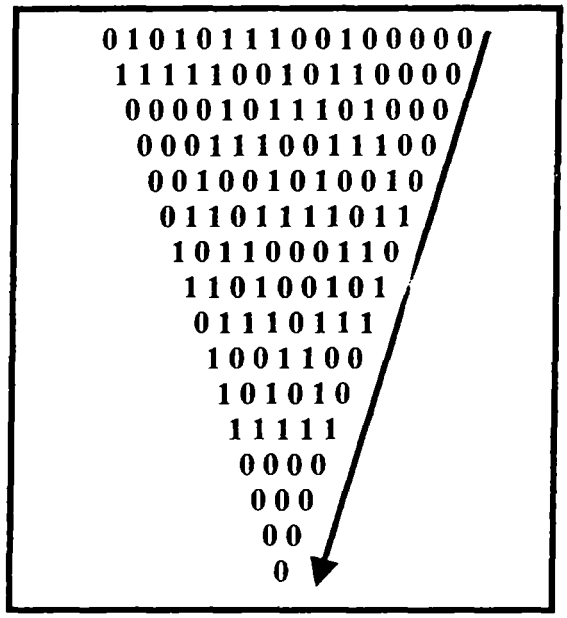


Figure 3.3.14
Construction of Target Block for Block 12 from the Generated Triangle

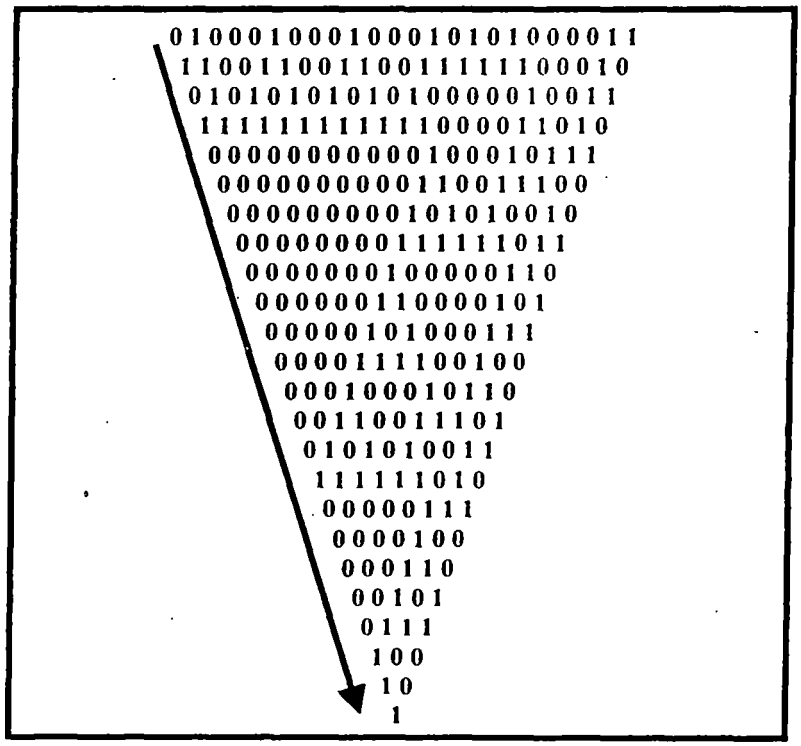


Figure 3.3.15
Construction of Target Block for Block 13 from the Generated Triangle

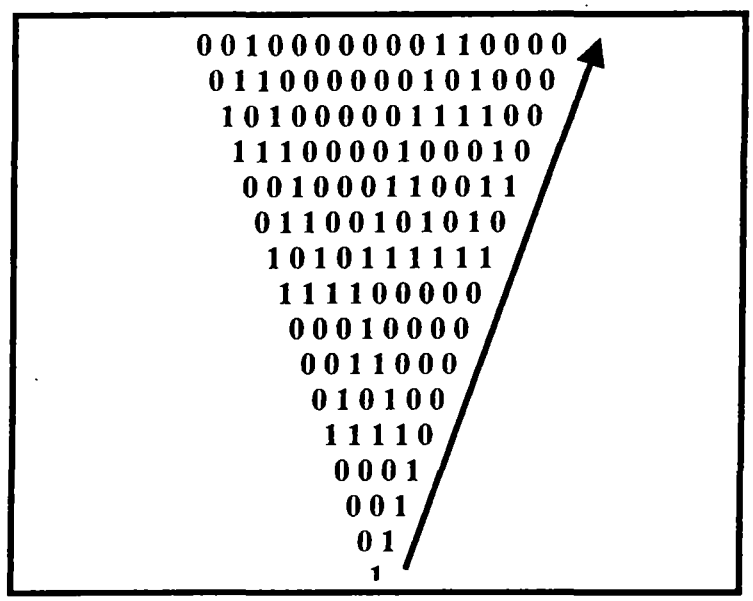


Figure 3.3.16
Construction of Target Block for Block 14 from the Generated Triangle

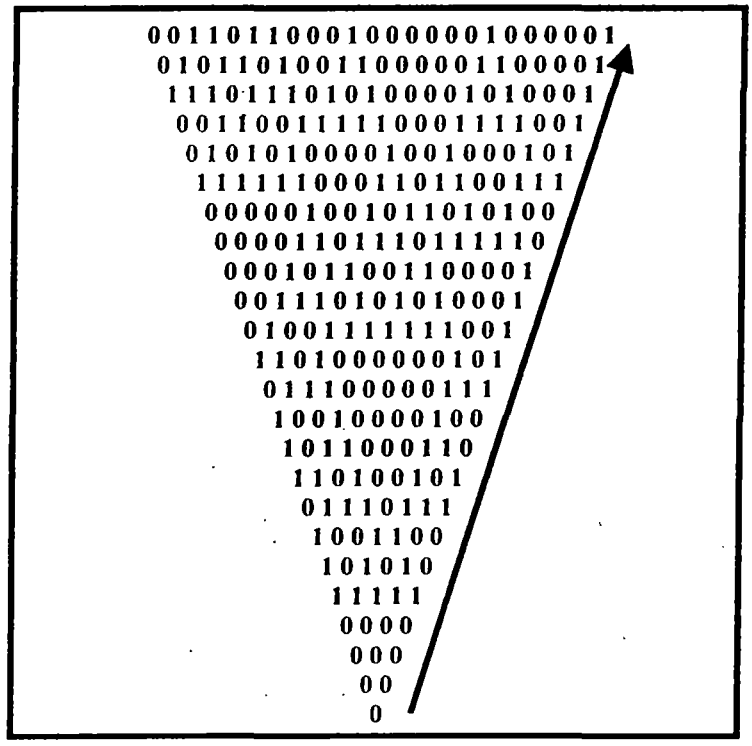


Figure 3.3.17
Construction of Target Block for Block 15 from the Generated Triangle

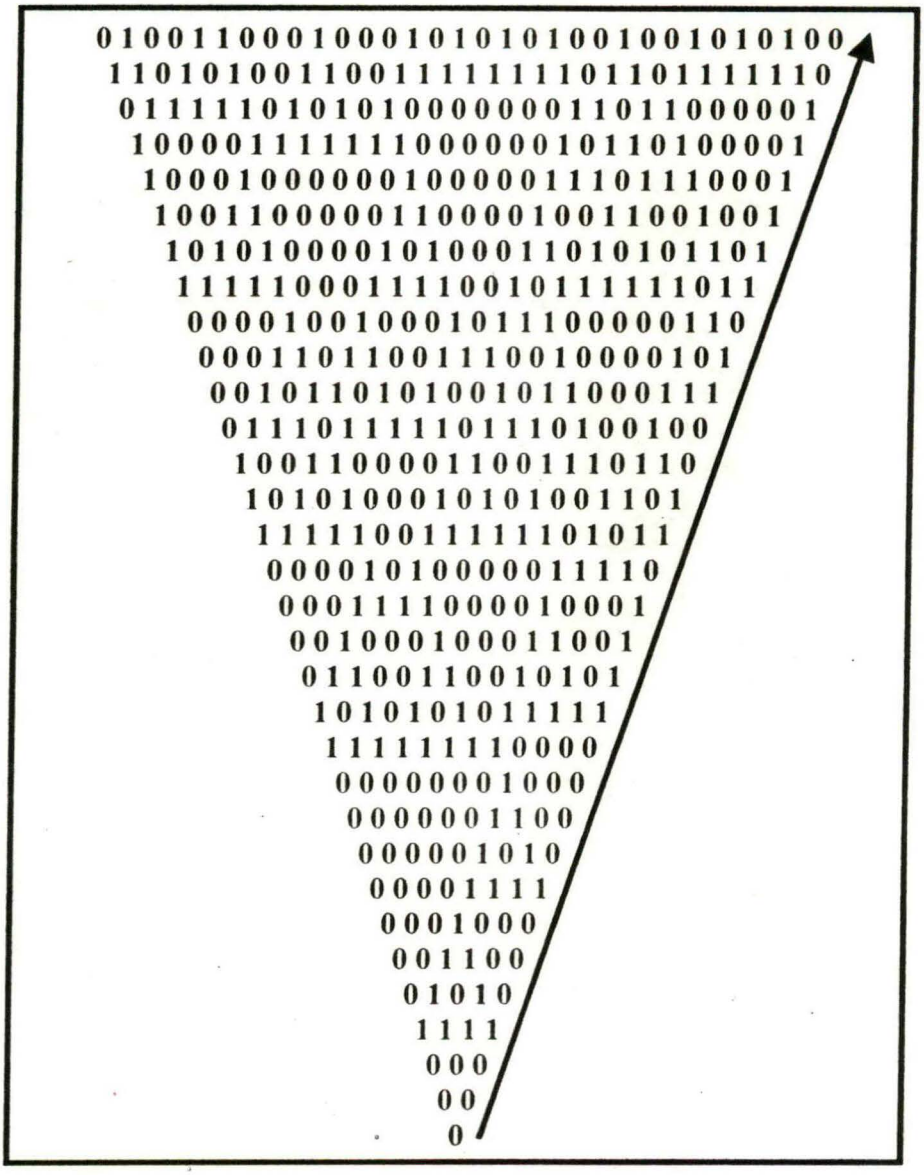


Figure 3.3.18
Construction of Target Block for Block 16 from the Generated Triangle

Table 3.3.4 enlists all target blocks formed through figures 3.3.3 to 3.3.18.

Table 3.3.4
All Target Blocks

Target Block 1	**01000001000100000000011000010000**
Target Block 2	**01011101000110100000011000000111**
Target Block 3	**00000110010111001100001111111100**
Target Block 4	**01011000011010001010100010000010**
Target Block 5	**001100110111010101110011**
Target Block 6	**100101101100001111110011**
Target Block 7	**011001100000111111110011**
Target Block 8	**010001001110100001001000**
Target Block 9	**001011100101011001011111011011100**
Target Block 10	**100010111111111100101101011011100**
Target Block 11	**110011110010001010100101111011101**
Target Block 12	**0000010110010000**
Target Block 13	**01010000000000100000111**
Target Block 14	**1111000001010000**
Target Block 15	**000010011001111100111111**
Target Block 16	**000100010000111101100110111111100**

By combining all target blocks shown in table 3.3.4, we obtain the following stream of bits as the encrypted stream:

0100000100010000000001100001000001011101000110100000011000000111000001100101110011000011111111000101100001101000101010001000001000110011011101010111001110010110110000111111100110110011000001111111100110100001001110100001000001011100101011001011110111011100100010111111111100101101010111001100111100100010101001011101110100000101100100000101000000000001000001111111000001010000000010011001111100111111100010001000011110110011011111100

Using "/" as the separator between two consecutive bytes, we present the encrypted stream like the following:

01000001/00010000/00000110/00010000/01011101/00011010/00000110/00000111/00000110/01011100/11000011/11111100/01011000/01101000/10101000/10000010/00110011/01110101/01110011/10010110/11000011/11110011/01100110/00001111/11110011/01000100/11101000/01001000/00101110/01010110/01011110/11011100/10001011/11111111/00101101/01011100/11001111/00100010/10100101/11011101/00000101/10010000/01010000/00000001/00000111/11110000/01010000/00001001/10011111/00111111/00010001/00001111/01100110/11111100

Table 3.3.5 converts bytes in the encrypted stream into characters.

Table 3.3.5
Converting Bytes into Characters

Byte	Character	Byte	Character	Byte	Character	Byte	Character
01000001	A	10000010	é	01011110	^	11110000	≡
00010000	▶	00110011	3	11011100	▬	01010000	P
00000110	♠	01110101	u	10001011	Ī	00001001	<Tab>
00010000	▶	01110011	s	11111111	<Blank>	10011111	*f*
01011101	]	10010110	û	00101101	-	00111111	?
00011010	→	11000011	├	01011100	\	00010001	◀
00000110	♠	11110011	≤	11001111	╧	00001111	☼
00000111	•	01100110	f	00100010	"	01100110	f
00000110	♠	00001111	☼	10100101	Ň	11111100	η
01011100	\ .	11110011	≤	11011101	■		
11000011	├	01000100	D	00000101	♣		
11111100	η	11101000	Φ	10010000	Ė		
01011000	X	01001000	H	01010000	P		
01101000	h	00101110	.	00000001	☻		
10101000	¿	01010110	V	00000111	.		

All characters from table 3.3.5 are combined sequentially to obtain the following:

"A▶▶▶→♠•♠\ ├η X h¿ é3 u s û ╞ f☼╞ D ΦH. V^╼╤ -\╧" Ň▬Ė P ●≡ P *f*? ◀☼f η"

This is the ciphertext to be transmitted corresponding to the plaintext taken for the implementation purpose. Figure 3.3.19 shows it in a pictorial manner.

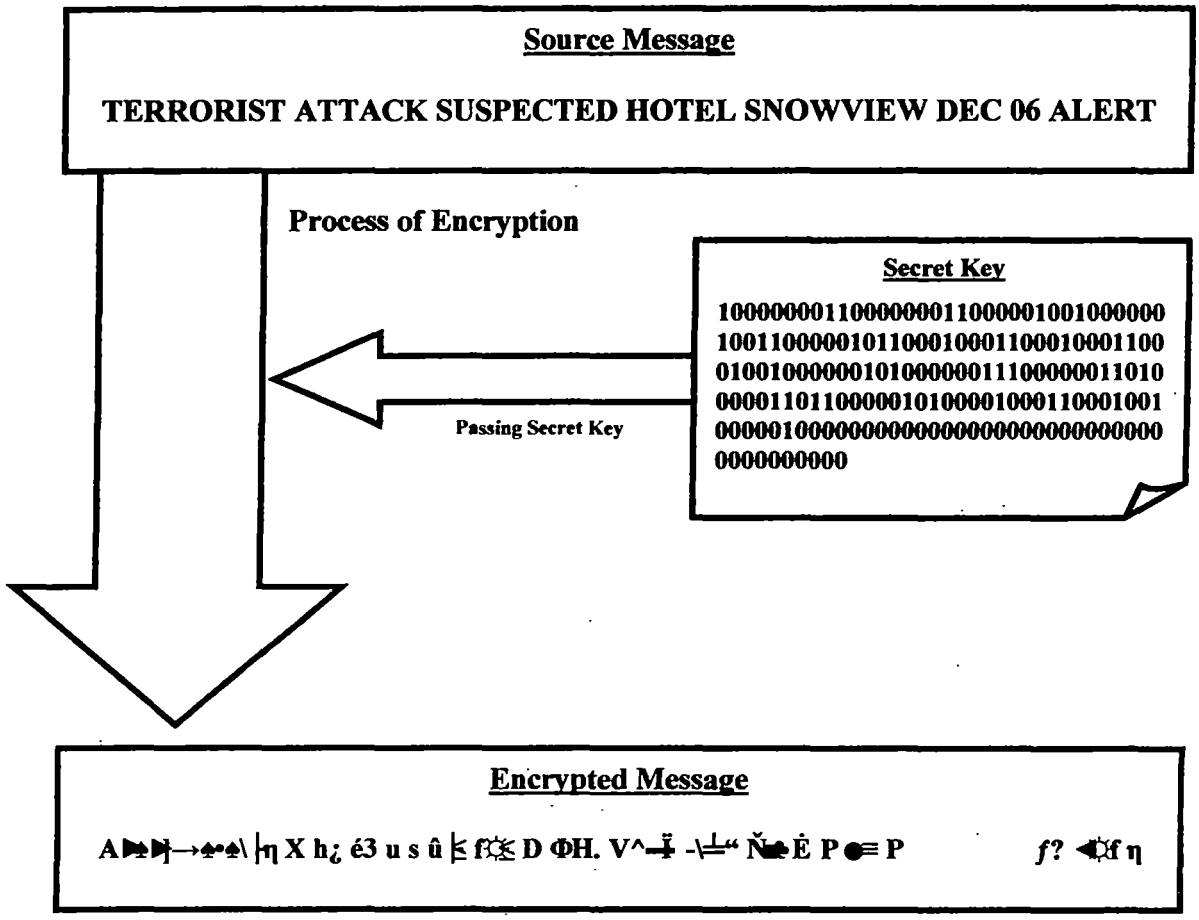


Figure 3.3.19
Generating Encrypted Message from Source using 180-Bit Secret Key

3.4 Results

For the purpose of implementation, a unique size of 64 bits has been considered, and for each such block, an asymmetric technique has been implemented.

Section 3.4.1 presents report of the encryption/decryption times and the Chi Square values. Section 3.4.2 presents result of frequency distribution tests. A comparative result with the RSA system is shown in section 3.4.3.

3.4.1 Result of Encryption/Decryption Time and Chi Square Value

Section 3.4.1.1 presents result for the .EXE files, section 3.4.1.2 presents result for the .COM files, section 3.4.1.3 presents result for the .DLL files, section 3.4.1.4

presents results for the .SYS files, and section 3.4.1.5 presents results for the .CPP files. For all the sections, the tabular representation of results include the encryption/decryption time, the numbers of operations required during encryption as well as decryption, and the chi square value [44, 48, 55, 56].

3.4.1.1 Result for *EXE* Files

Table 3.4.1.1.1 shows results. Ten files have been considered. The size of the files varies from 11611 bytes to 59398 bytes. The encryption time by using the proposed TE technique ranges from 0.714286 seconds to 5.439560 seconds. Using the TE technique for decryption, the decryption time ranges from 0.549451 to 2.692307 seconds. At the time of encryption, the total number of operations ranges from 74303488 to 448829696. During the process of decryption, the total number of operations ranges from 74041856 to 416387328. The Chi Square value between the source and the corresponding encrypted file ranges from 17993 to158628 with the degree of freedom ranging from 248 to 255.

Table 3.4.1.1.1
Result for .EXE Files for TE Technique

Source File	Encryption Time	Decryption Time	No. of Operations (During Encryption)	No. of Operations (During Decryption)	Chi Square Value	Degree of Freedom
TLIB.EXE	1.428571	1.703297	76004096	76004096	128674	255
MAKER.EXE	2.252747	2.692307	197401344	197401344	158628	255
UNZIP.EXE	0.879121	1.098901	244495104	244495104	17993	255
RPPO.EXE	1.318681	1.593407	316443904	316836352	55351	255
PRIME.EXE	1.373626	1.[illegible]8242	393756160	392709632	62986	255
TCDEF.EXE	1.043956	0.549451	448829696	416387328	33539	254
TRIANGLE.EXE	5.439560	1.703297	74303488	74041856	57961	255
PING.EXE	0.934066	1.098901	124536832	124275200	79747	248
NETSTAT.EXE	1.208791	1.483516	191514624	191252992	110332	255
CLIPBRD.EXE	0.714286	0.879121	229189632	228928000	65560	255

The graphical relationship between the source file size and the encryption/decryption time for .EXE files is shown in figure 3.4.1.1.1. It is observed in the graph that there exists a tendency that the encryption/decryption time varies linearly with the source file size, although there also exist a few exceptions.

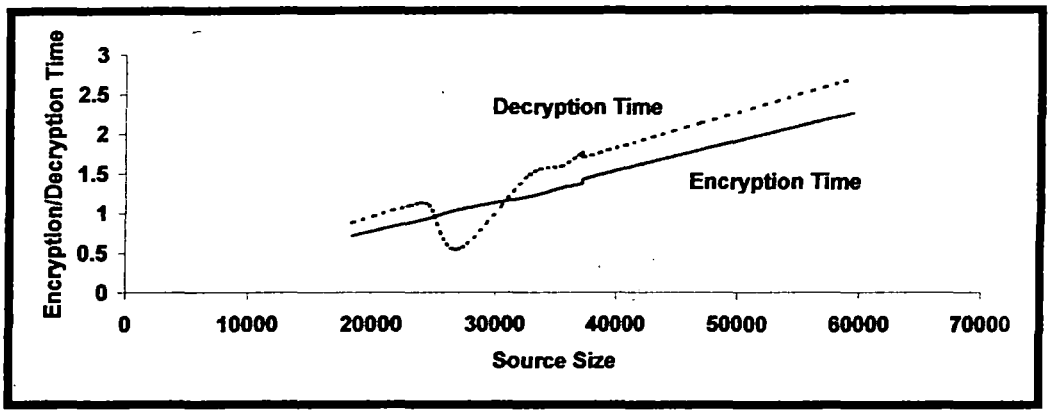


Figure 3.4.1.1.1
Relationship between Source Size and Encryption/Decryption Time for .EXE Files in TE Technique

3.4.1.2 Result for *COM* Files

Table 3.4.1.2.1 shows results. Ten files have been considered. The size of the files varies from 5239 bytes to 29271 bytes. The encryption time by using the proposed TE technique ranges from 0.164835 seconds to 0.989011 seconds. Using the TE technique for decryption, the decryption time ranges from 0.274725 to 1.373626 seconds. At the time of encryption, the total number of operations ranges from 40160512 to 361052160. During the process of decryption, the total number of operations also ranges from 40160512 to 361052160. The Chi Square value between the source and the corresponding encrypted file ranges from 6451 to 63314 with the degree of freedom ranging from 230 to 255.

Table 3.4.1.2.1
Result for .COM Files for TE Technique

Source File	Encryption Time	Decryption Time	No. of Operations (During Encryption)	No. of Operations (During Decryption)	Chi Square Value	Degree of Freedom
EMSTEST.COM	0.769231	0.934066	40160512	40160512	33211	255
THELP.COM	0.439560	0.549451	62791680	62791680	28716	250
WIN.COM	0.989011	1.153846	113417472	113417472	47792	252
KEYB.COM	0.769231	0.934066	154101248	154101248	48939	255
CHOICE.COM	0.164835	0.274725	164697344	164697344	9254	232
DISKCOPY.COM	0.824176	1.043956	209567232	209567232	51815	254
DOSKEY.COM	0.549451	0.769231	241224704	241224704	39119	253
MODE.COM	1.098901	1.373626	301007616	301007616	63314	255
MORE.COM	0.384615	0.549451	322330624	322330624	6451	230
SYS.COM	0.714286	0.879121	361052160	361052160	43706	254

Figure 3.4.1.2.1 shows how the decryption time changes with the encryption time for .COM files. This graph establishes the fact that the time complexity during the process of encryption varies almost linearly with that during the process of decryption.

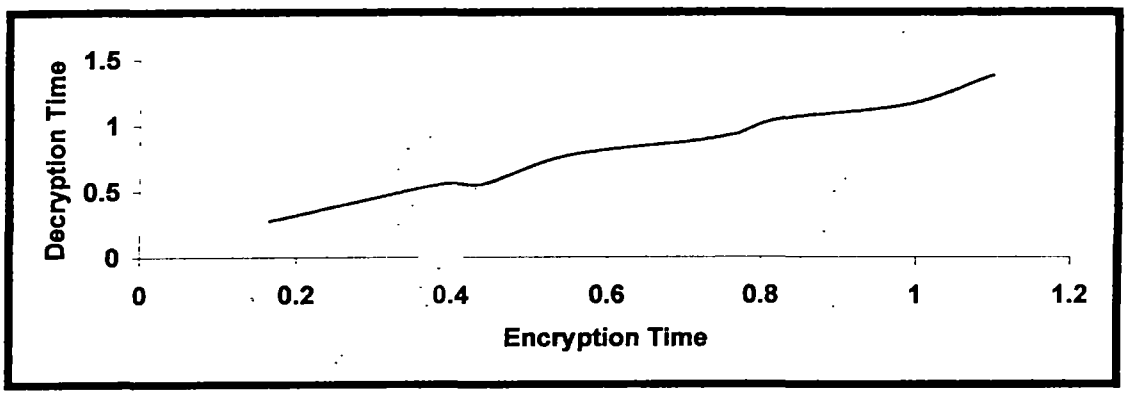


Table 3.4.1.2.1
Relationship between Encryption Time and Decryption Time for .COM Files in TE Technique

3.4.1.3 Result for *DLL* Files

Table 3.4.1.3.1 shows results. Ten files have been considered. The size of the files varies from 3216 bytes to 58368 bytes. The encryption time by using the proposed TE technique ranges from 0.164835 seconds to 2.142857 seconds. Using the TE technique for decryption, the decryption time ranges from 0.164835 seconds to 2.637362 seconds. At the time of encryption, the total number of operations ranges from 66977792 to 526403584. During the process of decryption, the total number of operations also ranges from 66977792 to 526403584. The Chi Square value between the source and the corresponding encrypted file ranges from 8383 to 205511 with the degree of freedom ranging from 217 to 255.

Table 3.4.1.3.1
Result for .DLL Files for TE Technique

Source File	Encryption Time	Decryption Time	No. of Operations (During Encryption)	No. of Operations (During Encryption)	Chi Square Value	Degree of Freedom
SNMPAPI.DLL	1.208791	1.483516	66977792	66977792	70479	253
KPSHARP.DLL	1.153846	1.483516	131862528	131862528	203671	254
WINSOCK.DLL	0.769231	0.989011	175816704	175816704	105418	252
SPWHPT.DLL	1.263736	1.428571	242794496	242794496	182841	255
HIDCI.DLL	0.164835	0.164835	249335296	249335296	8383	217
PFPICK.DLL	2.142857	2.637362	368639488	368639488	153199	255
NDDEAPI.DLL	0.549451	0.659341	397288192	397288192	56969	249
NDDENB.DLL	0.439560	0.494505	419657728	419657728	60992	251
ICCCODES.DLL	0.769231	0.989011	462565376	462565376	168116	252
KPSCALE.DLL	1.153846	1.428571	526403584	526403584	205511	255

The linear relationship between the source size and the encryption time for .DLL files is shown in figure 3.4.1.3.1.

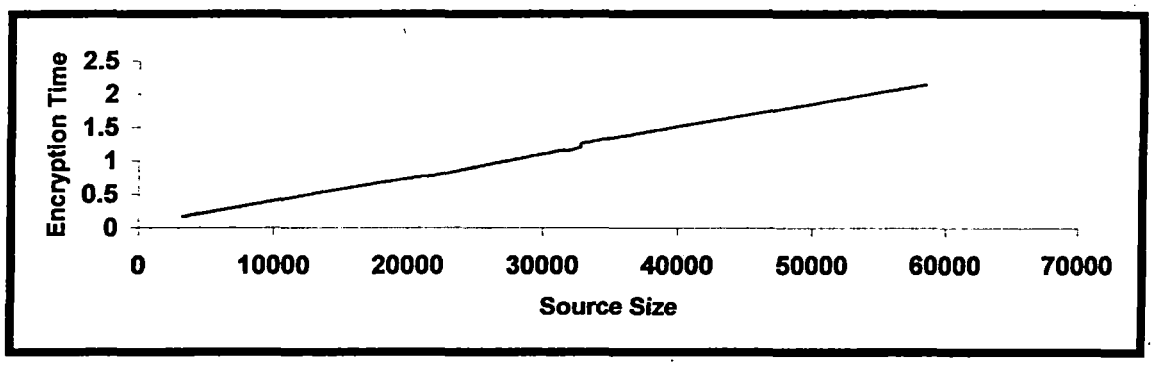


Table 3.4.1.3.1
Linear Relationship between Source Size and Encryption Time for .DLL Files in TE Technique

3.4.1.4 Result for *SYS* Files

Table 3.4.1.4.1 shows results. Ten files have been considered. The size of the files varies from 1105 bytes to 33191 bytes. The encryption time by using the proposed TE technique ranges from 0.054945 seconds to 1.263736 seconds. Using the TE technique for decryption, the decryption time ranges from 0.109890 seconds to 1.428571 seconds. At the time of encryption, the total number of operations ranges from 38590720 to 222648832. During the process of decryption, the total number of operations also ranges from 38590720 to 222648832. The Chi Square value between the source and the corresponding encrypted file ranges from 859 to 126367 with the degree of freedom ranging from 165 to 255.

Table 3.4.1.4.1
Result for .SYS Files for TE Technique

Source File	Encryption Time	Decryption Time	No. of Operations (During Encryption)	No. of Operations (During Decryption)	Chi Square Value	Degree of Freedom
HIMEM.SYS	1.263736	1.428571	67762688	67762688	63553	255
RAMDRIVE.SYS	0.494505	0.549451	93533440	93533440	16452	241
USBD.SYS	0.714286	0.824176	38590720	3859072	84593	255
CMD640X.SYS	0.879121	1.098901	88824064	88824064	65706	255
CMD640X2.SYS	0.769231	0.934066	131470080	131470080	58201	255
REDBOOK.SYS	0.219780	0.219780	142981888	142981888	21052	230
IFSHLP.SYS	0.109890	0.164835	150438400	150438400	6488	237
ASPI2HLP.SYS	0.054945	0.109890	152662272	152662272	859	165
DBLBUFF.SYS	0.109890	0.109890	157894912	157894912	3190	215
CCPORT.SYS	1.153846	1.373626	222648832	222648832	126367	255

Figure 3.4.1.4.1 graphically compares the encryption time and the decryption time for .SYS files, in which the white part stands for the encryption time, and the black portion stands for the decryption time. For each file, the corresponding pillar stands for the total time required for the encryption and the decryption purposes. Out of each of the pillars, almost half is occupied by the white part and the remaining by the black part, which establishes the fact that the encryption time and the decryption time are almost equal to each other.

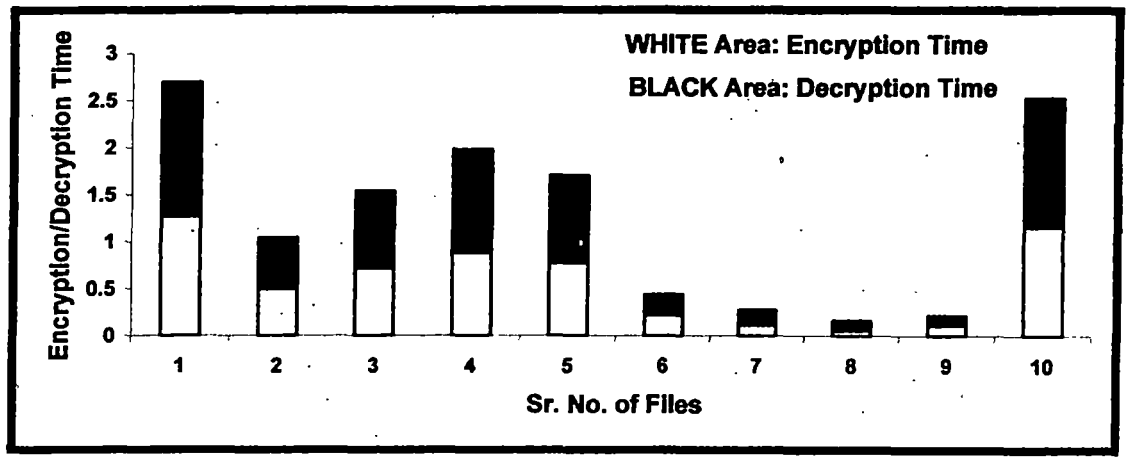


Figure 3.4.1.4.1
Comparison between Encryption Time and Decryption Time for .SYS Files in TE Technique

3.4.1.5 Result for *CPP* Files

Table 3.4.1.5.1 shows results. Ten files have been considered. The size of the files varies from 4071 bytes to 14557 bytes. The encryption time by using the proposed TE technique ranges from 0.054945 seconds to 0.164835 seconds. Using the TE technique for decryption, the decryption time ranges from 0.164835 seconds to 1.428571 seconds. At the time of encryption, the total number of operations ranges from 34142976 to 246980608. During the process of decryption, the total number of operations also ranges from 34142976 to 246980608. The Chi Square value between the source and the corresponding encrypted file ranges from 1305 to 144422 with the degree of freedom ranging from 69 to 90.

Table 3.4.1.5.1
Result for .CPP Files for TE Technique

Source File	Encryption Time	Decryption Time	No. of Operations (During Encryption)	No. of Operations (During Decryption)	Chi Square Value	Degree of Freedom
BRICKS.CPP	0.714286	0.769231	34142976	34142976	68626	88
PROJECT.CPP	1.208791	1.428571	99812608	99812608	144422	90
ARITH.CPP	0.384615	0.439560	119304192	119304192	15466	77
START.CPP	0.549451	0.659341	148999424	148999424	47550	88
CHARTCOM.CPP	0.494505	0.659341	177778944	177778944	45205	84
BITIO.CPP	0.164835	0.164835	186020352	186020352	8387	70
MAINC.CPP	0.164835	0.219780	195439104	195439104	7916	83
TTEST.CPP	0.054945	0.054945	197924608	197924608	1305	69
DO.CPP	0.549451	0.659341	227489024	227489024	48061	88
CAL.CPP	0.384615	0.439560	246980608	246980608	17336	77

The linear relationship between the encryption time and the decryption time for .CPP files is shown in figure 3.4.1.5.1

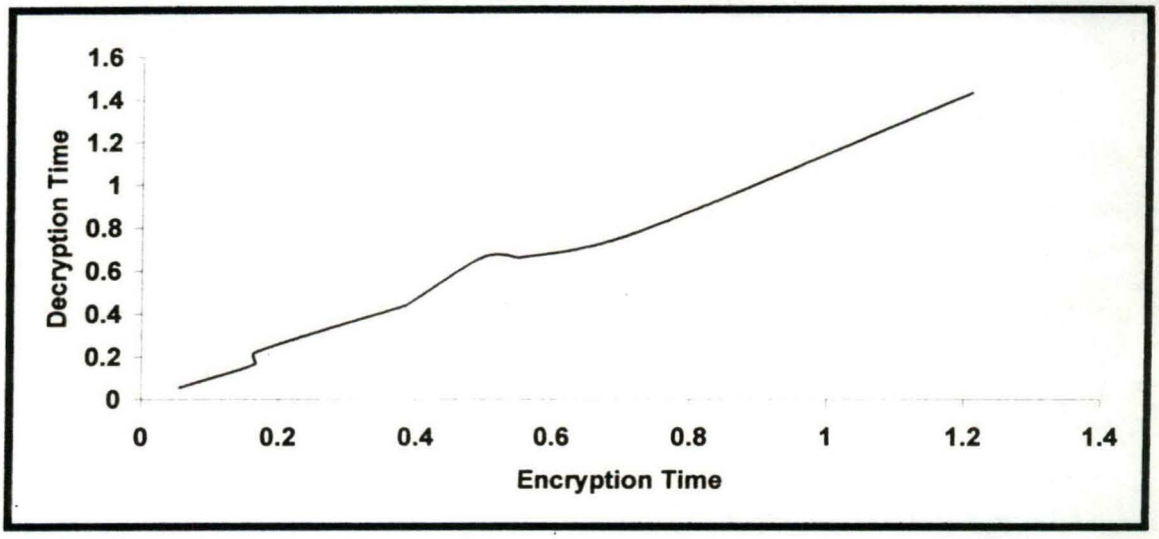


Figure 3.4.1.5.1
Linear Relationship between Encryption Time and Decryption Time for .CPP Files

3.4.2 Result of Frequency Distribution Tests

For each sample file, the frequency of each character is computed and it is compared with that in the corresponding encrypted file. It is seen for all cases that the characters in the encrypted files are well distributed. This shows that the technique proposed here is quite compatible with available techniques. The bars in red color represent frequencies of characters in the encrypted file and those in blue color represent frequencies of characters in the source file. The frequencies of characters in encrypted files are evenly distributed. It can be said that the source and the corresponding encrypted file are heterogeneous in nature. Therefore it can be interpreted that through the proposed technique, a good quality of encryption is obtained.

Figure 3.4.2.1 to figure 3.4.2.5 shows five of such results obtained through the frequency distribution tests, taking one from each category of files.

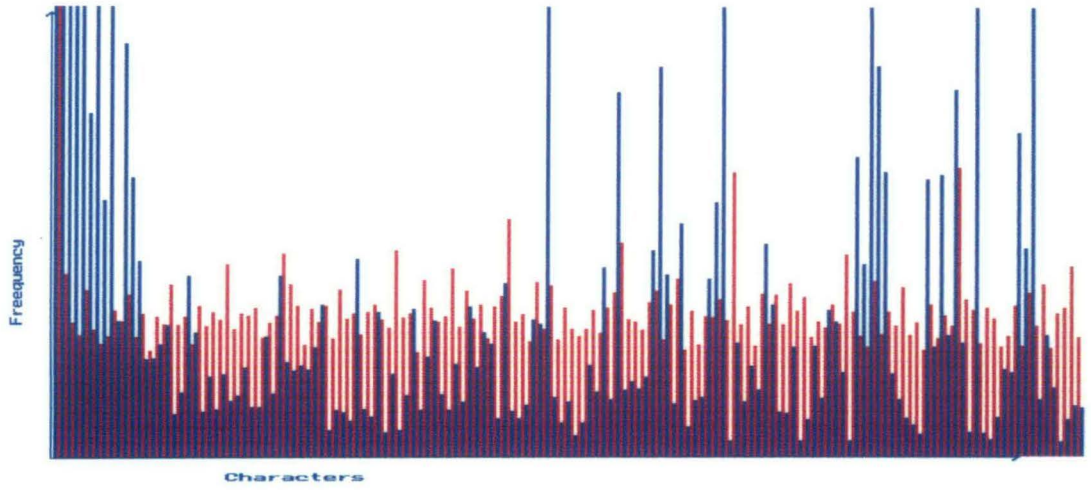


Figure 3.4.2.1
Frequency Distribution of Characters between
TRIANGLE.EXE and Encrypted FOX1.EXE

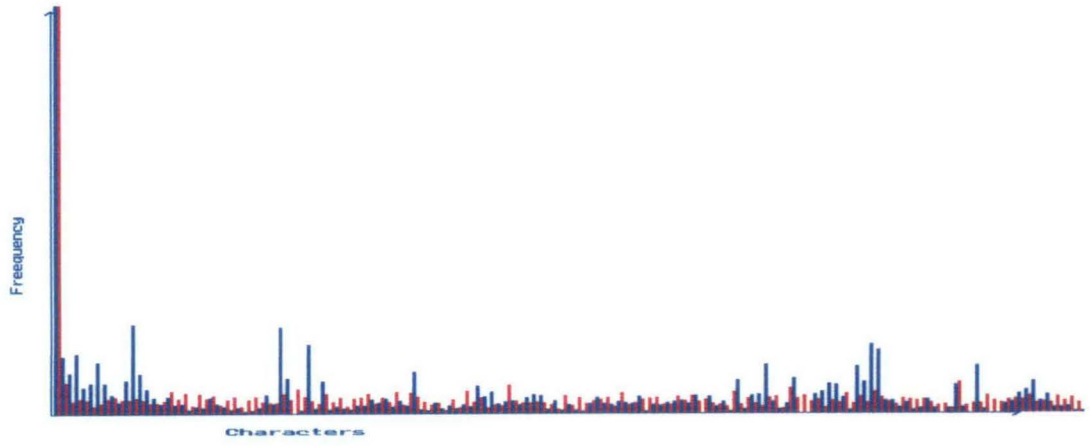


Figure 3.4.2.2
Frequency Distribution of Characters between
MORE.COM and Encrypted FOX1.COM

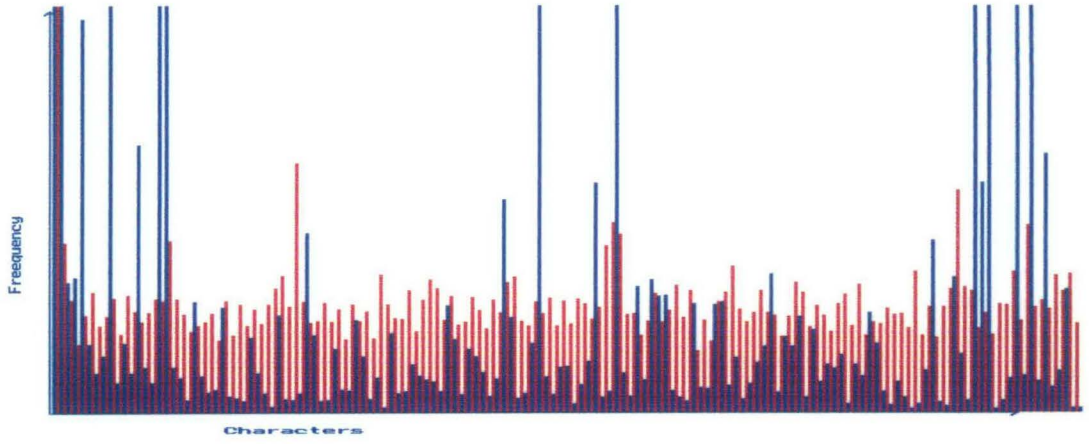


Figure 3.4.2.3
Frequency Distribution of Characters between KPSCALE.DLL and Encrypted FOX1.DLL

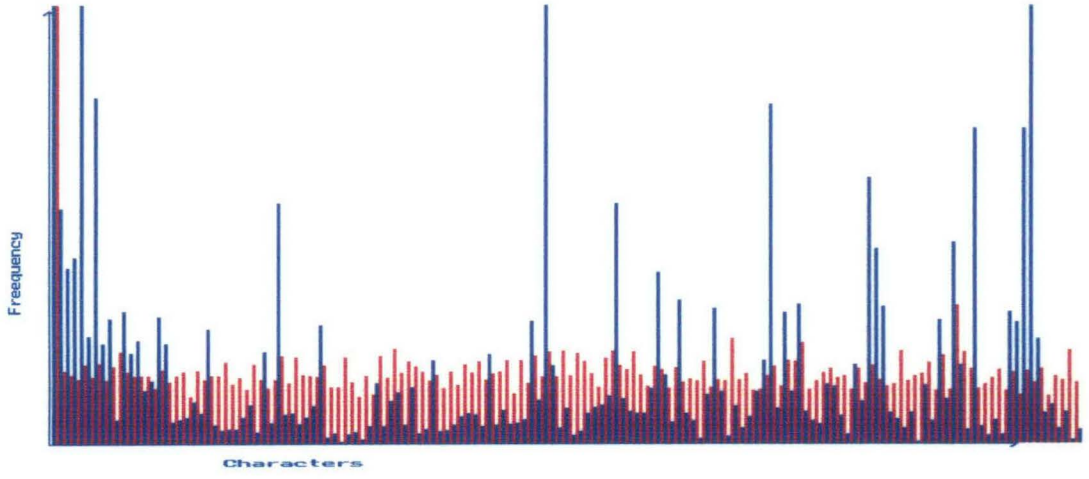


Figure 3.4.2.4
Frequency Distribution of Characters between CMD640X2.SYS and Encrypted FOX1.SYS

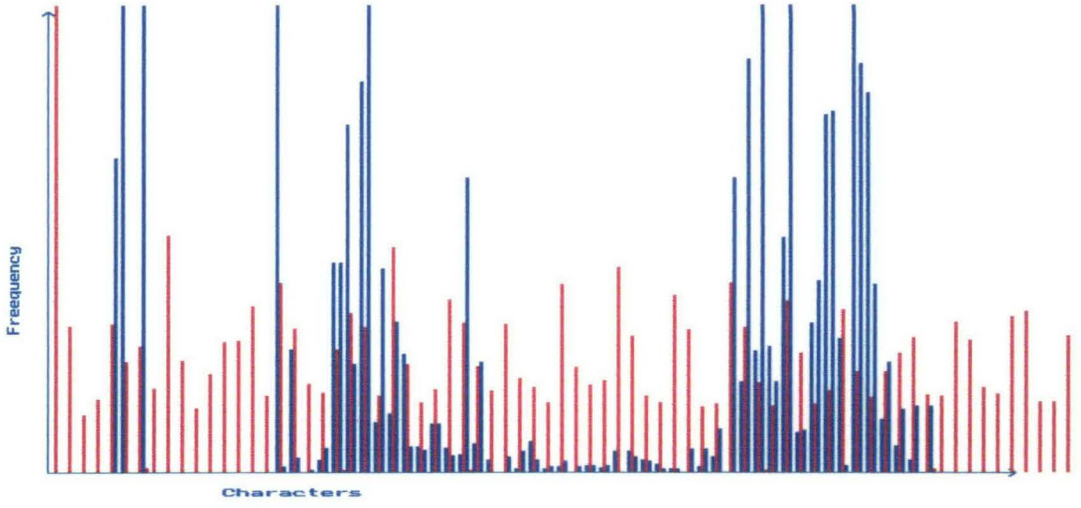


Figure 3.4.2.5
Frequency Distribution of Characters between
DO.CPP and Encrypted FOX1.CPP

3.4.3 Comparison of TE with RSA Technique

On the basis of the Chi Square values between .CPP sample files and the corresponding encrypted files, the proposed TE technique has been compared with the existing RSA system. Ten sample files of different sizes are taken. Chi Square values are computed for both TE and RSA techniques. Both techniques show very high values in Chi Square tests. This proves the high degree of non-homogenity between source files and corresponding encrypted files [7, 8]. Table 3.4.3.1 enlists these comparative results.

Table 3.4.3.1
Comparison of Chi Square Values between TE and RSA

Source File (1)	Source Size	Encrypted File using TE Technique (2)	Encrypted File using RSA Technique (3)	Chi Square Value between (1) and (2)	Chi Square Value between (1) and (3)	Degree of Freedom
BRICKS.CPP	16723	*FOX1.CPP*	*CPP1.CPP*	68626	200221	88
PROJECT.CPP	32150	*FOX2.CPP*	*CPP2.CPP*	144422	197728	90
ARITH.CPP	9558	*FOX3.CPP*	*CPP3.CPP*	15466	273982	77
START.CPP	14557	*FOX4.CPP*	*CPP4.CPP*	47550	49242	88
CHARTCOM.CPP	14080	*FOX5.CPP*	*CPP5.CPP*	45205	105384	84
BITIO.CPP	4071	*FOX6.CPP*	*CPP6.CPP*	8387	52529	70
MAINC.CPP	4663	*FOX7.CPP*	*CPP7.CPP*	7916	4964	83
TTEST.CPP	1257	*FOX8.CPP*	*CPP8.CPP*	1305	3652	69
DO.CPP	14481	*FOX9.CPP*	*CPP9.CPP*	48061	655734	88
CAL.CPP	9540	*FOX10.CPP*	*CPP10.CPP*	17336	216498	77

Figure 3.4.3.1 graphically shows the comparison.

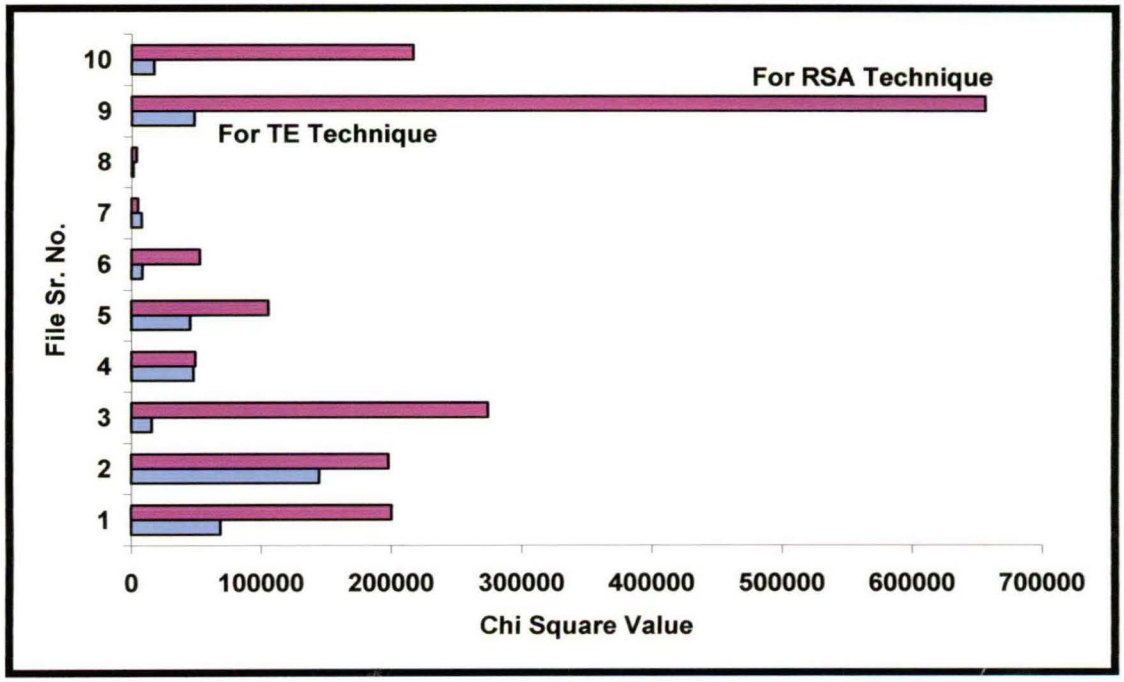


Figure 3.4.3.1
Graphical Comparison of Results of Chi Square Tests for Proposed TE and Existing RSA

3.5 Analysis and Conclusion including Comparison with RPSP

On the basis of the practical implementation on a sample set of 50 real files, it is observed that the process of encryption/decryption using the proposed TE techniques requires less time on the average than that using the proposed RPSP technique, presented in chapter 2. The average Chi Square value is observed to be higher in case of the TE technique than the RPSP technique [48, 52]. Table 3.5.1 show this comparative result. In chapter 10, graphical comparisons have been drawn.

Table 3.5.1
Average Encryption/Decryption Time and Chi Square Value obtained in RPSP and TE Techniques

Proposed Technique	Average Encryption Time	Average Decryption Time	Average Chi Square Value	Average Degree of Freedom
RPSP	8.75713800	8.73955200	10701.70	214
TE	0.86703290	0.94175818	64188.04	

The TE policy of encryption is a strong scheme due to the existence of the multiple numbers of options for encrypting each block. Choosing varying block lengths and varying options for different blocks help in requiring a long key space. One such key structure strictly following a set of encryption protocol is proposed in section 8.2.2 in chapter 8.

The encryption/decryption time largely depends on the size of the source file, and the encrypted files are highly non-homogeneous with the respective source files with the encrypted characters are well distributed to almost nullify the chance through cryptanalysis to break the cipher.

By applying flexibility in choosing sizes of blocks and options for each of the blocks, the policy of the TE encryption is capable of producing the information security of a highly satisfactory level. Also, it helps in enhancing performance while taking part in the cascaded approach of encryption.