Encryption Through Triangulation Encryption (TE)

To be submitted by

ADITYA DAS

In partial fulfilment for the degree of
Master Of Computer Science & Engineering

Under the supervision of

Prof. Jyotsna Kumar Mandal

Department of Computer Science and Engineering
University of Kalyani

Department of Computer Science and Engineering
University of Kalyani
Kalyani, Nadia, Pin : 741235, West Bengal, India

January 2025

University of Kalyani

Faculty Of Engineering, Technology And Management

Prof. Jyotsna Kumar Mandal
Professor
Department of Computer Science and Engineering ,
University of Kalyani

Kalyani -741235
Nadia, West Bengal, India
Phone : +91 -33 -5809617
(Direct) +91 -33 -25828750/4378
Extn. -304/256/225
Fax : - +91-33-5809617

Supervisor's Certificate

This is to certify that the partial fulfilment of the project report entitled **"Encryption Through Triangulation Encryption (TE)"** submitted by Mr Aditya Das, bearing Registration Number 5080003 of 2023-2024 and Roll No: 90/CSE/230001, a student of MTECH in Department of Computer Science and Engineering under The University of Kalyani, is based upon his own work under my supervision and that neither his project nor any part of the project has been submitted for any degree or diploma or any other academic award anywhere before. I wish him all the success.

Place : Kalyani
Date : 17/01/2025

_____

Prof. Jyotsna Kumar Mandal
Professor
Department of Computer Science and Engineering
Faculty Of Engineering, Technology And Management
University of Kalyani, Kalyani : 741235, Nadia, West Bengal, India

Statutory Declarations

Name of the Candidate: Aditya Das

Title of the Project: Encryption Through Triangulation Encryption (TE)

Degree: Masters of Computer Science & Technology (M.tech)

Name of the Guide: Prof. Jyotsna Kumar Mandal

Registration Number: 5080003 of 2023-2024

Roll Number: 90/CSE/230001

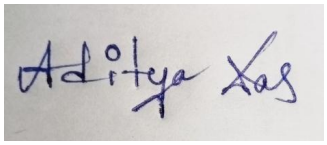
Place of Project: Department of Computer Science and Engineering,
University of Kalyani,
Kalyani, Nadia, West Bengal, India

Declaration by the Student

I hereby declare that the work reported in the M.Tech Project entitled **"Encryption Through Triangulation Encryption (TE)"** is an authentic record of my work carried out under the supervision of Prof. Jyotsna Kumar Mandal. I have not submitted this work elsewhere for any other degree or diploma.

Place : Kalyani

Date : 17/01/2025



Signature of Student

ACKNOWLEDGEMENT

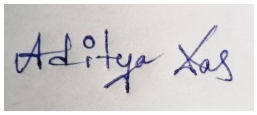
I would like to express my deep sense of gratitude and indebtedness to the many individuals who contributed to the success of my project. Foremost, I am fortunate to have had the guidance and support of my supervisor, Prof. Jyotsna Kumar Mandal. His uncountable advice and encouragement played a pivotal role in the systematic completion of my project, and I am confident that his influence will resonate throughout my career.

I extend my thanks to the Head of the Department, Prof. Kalyani Mali, as well as Prof. Anirban Mukbopadhyay, Prof. Utpal Biswas, Prof. Priya Ranjan Sinha Mahapatra, Dr. Debabrata Sarddar, Mr. Sukanta Majumdar, Mr Jaydeep Paul and Mrs. Shrabanti Kundu from the Department of Computer Science and Engineering at the University of Kalyani. Their support and cooperation were invaluable.

I am grateful to the entire Department of Computer Science and Engineering at the University of Kalyani for their assistance and collaborative efforts.

During the course of my project, I received tremendous support from my friends and classmates. My sincerest thanks go to them for their invaluable assistance.

No success would be complete without acknowledging the role of my parents. Their unwavering support and encouragement have been the driving force behind my education and achievements. They made it their life's mission to ensure I stayed focused on my goals. With their blessings and wise words, I found the strength to overcome challenges. I deeply appreciate their foresight and simplicity, which have been instrumental in helping me achieve my aspirations.



(Aditya Das)

Content

1. Introduction	**8**
2. The Scheme	**9**
2.1 Example	**9 - 14**
3. Implementation	**15 - 26**
4. Chi square Calculation	**27**
5. Results	**28**
5.1 Results of .CPP Files	**29 - 30**
5.2 Results of .SYS Files	**31 - 32**
5.3 Results of .TXT Files	**33 - 34**
5.4 Results of .DLL Files	**35 - 36**
5.5 Results of .EXE Files	**37 - 38**
6. TE with Session Key	
6.1. Generating the Session Key	**39**
6.2. File Dividing Logic	**40 - 41**
6.3. Implementation	**42**
7. Results of TE with Session Key	**43**
7.1. Results on .TXT Files	**44 - 45**
7.2. Results on .EXE Files	**46 - 47**
7.3. Results on .CPP Files	**48 - 49**
7.4. Results on .SYS Files	**50 - 51**
7.5. Results on .DLL Files	**52 - 53**

8. Comparison between Encryption and Decryption Time of TE and TE using Session Key 54

8.1.1. Comparison between Encryption Time of TE and TE using Session Key of .TXT files
8.1.2. Comparison between Decryption Time of TE and TE using Session Key of .TXT files
8.2.1. Comparison between Encryption Time of TE and TE using Session Key of .EXE files
8.2.2. Comparison between Decryption Time of TE and TE using Session Key of .EXE files
8.3.1. Comparison between Encryption Time of TE and TE using Session Key of .CPP files
8.3.2. Comparison between Decryption Time of TE and TE using Session Key of .CPP files
8.4.1. Comparison between Encryption Time of TE and TE using Session Key of .SYS files
8.4.2. Comparison between Decryption Time of TE and TE using Session Key of .SYS files
8.5.1. Comparison between Encryption Time of TE and TE using Session Key of .DLL files
8.5.2. Comparison between Decryption Time of TE and TE using Session Key of .DLL files

9. NIST Test Report 60 - 68

10. Conclusion 69

11. Future Works 70

12. References 71 - 73

1. Introduction :

(TE)Triangular Encryption Technique, is 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, is the use of the Boolean operation. Also, unlike the RPSP technique, 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 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.

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 an enhancement of security can be done by allowing block sizes to be different because it makes the key length large enough, thereby almost nullifying the chance of cryptanalysis to break the cipher. The entire scheme includes several parts in it. First of all is the formation of the triangle, different options that are available to form the target block from the triangle generated gives the processes for decryption to generate the source block from a target block.

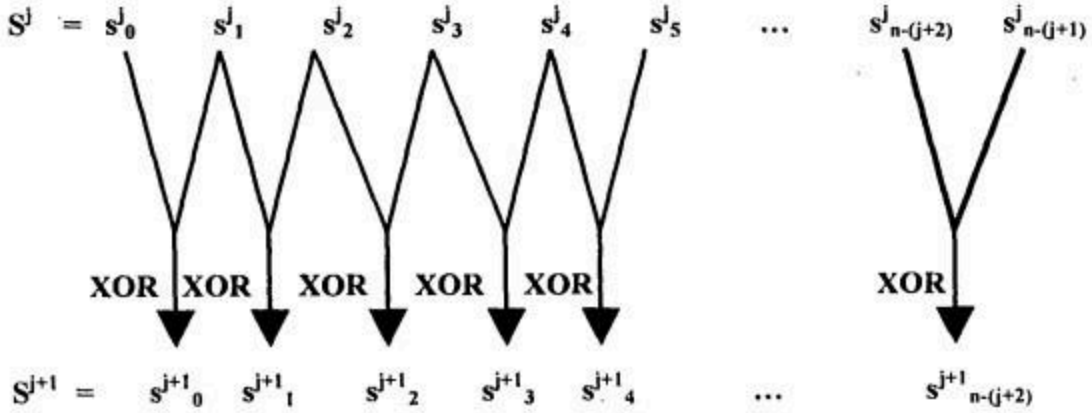
2.1 Example :

- **Formation of Triangle**

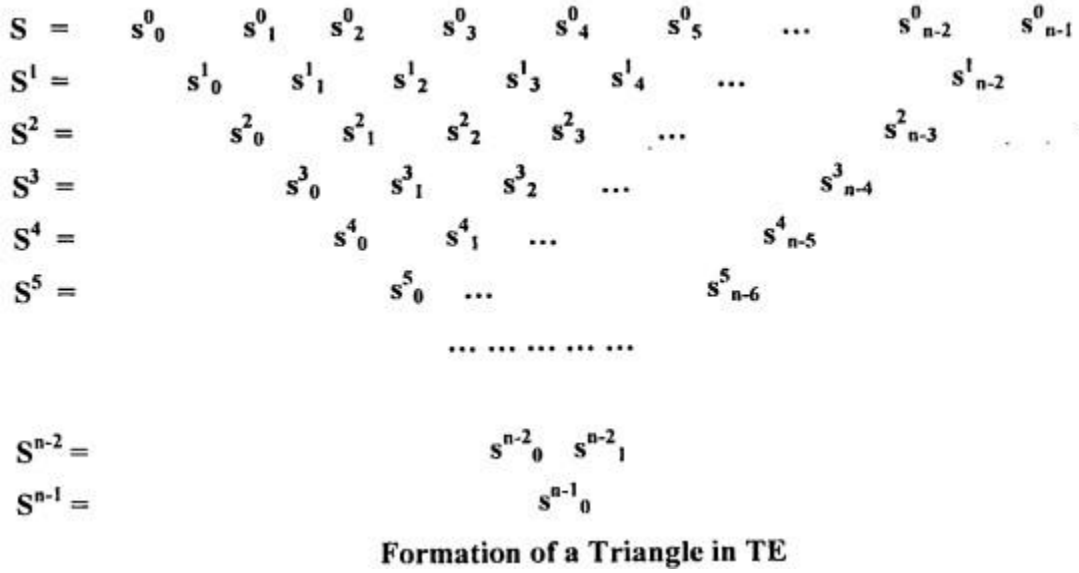
Consider a block $S = s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4\ s^0_5 \ldots s^0_{n-2}\ s^0_{n-1}$ of size n bits, where $s^0_i = 0$ or 1 for $0 <= i <= (n-1)$.

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



Generation of an Intermediate Sub-Stream in TE



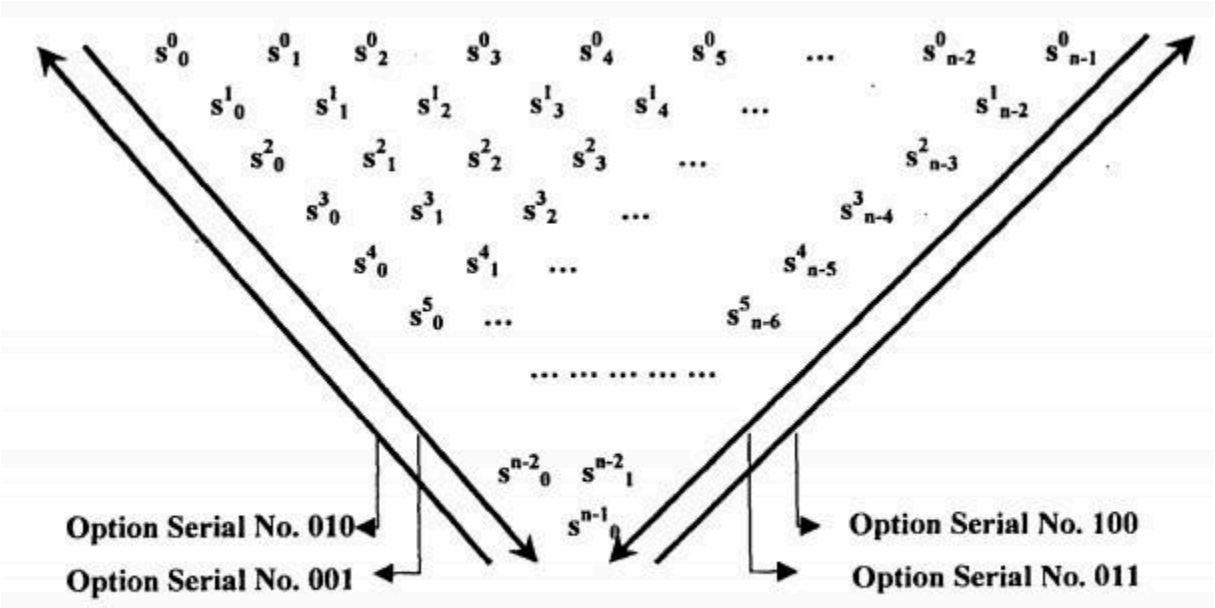
Formation of a Triangle in TE

- **Options for Forming Target Blocks from Triangle**

Corresponding to figure 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

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



Diagrammatic Representation of Options for choosing Target Block from Triangle

- **Generating Source Block from a Target Block**

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

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

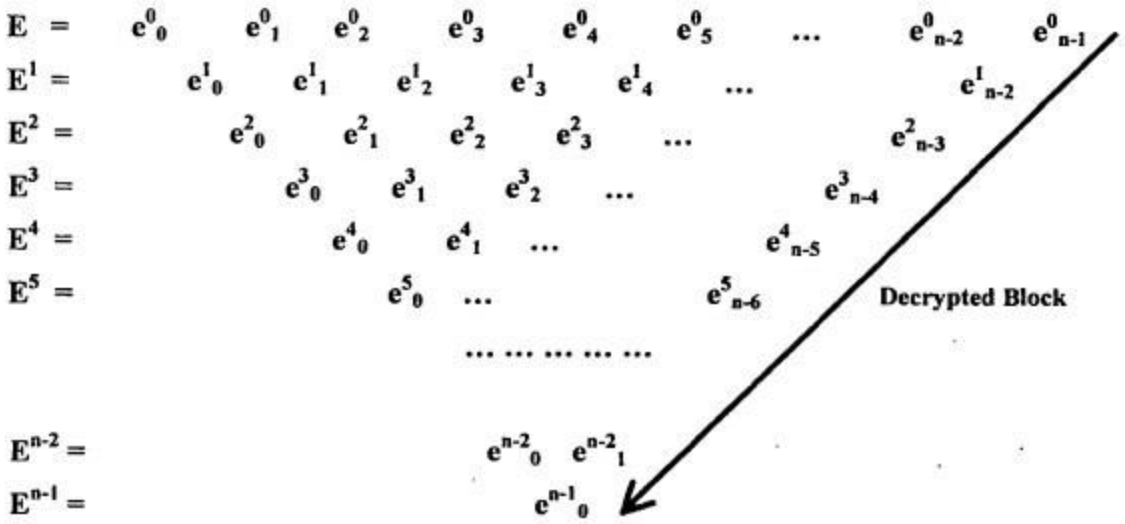
As is shown in table corresponding to option 001 and option 100, the processes of generating the source block are the same, which means, forming the triangle and picking bits in the same manner one by one.

The section 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

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 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 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. after 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}.

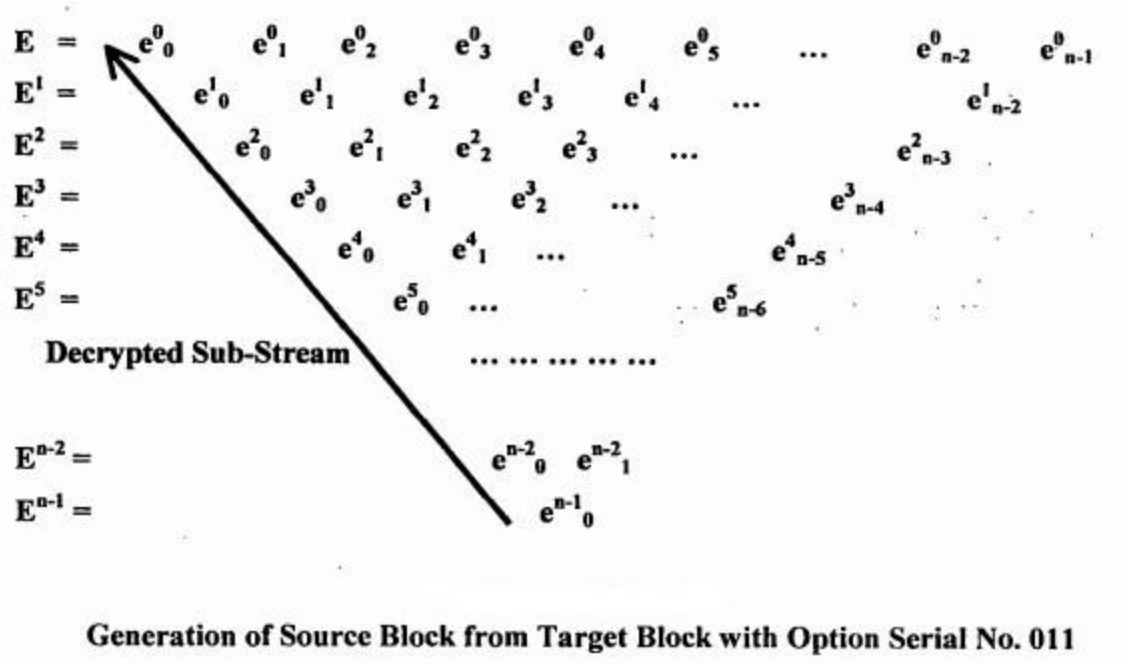


Generation of Source Block from Target Block with Option Serial No. 010

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

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

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 , 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. after 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}.



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

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 the proposed structure of the corresponding 180-bit key has been proposed.

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

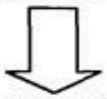
Different Considerations for Encryption using TE

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 and the key will be as shown in below

The Secret Key
(Using "/" as Separators)

100000/001/100000/001/100000/100/100000/010/011000/001/011000/100/011000/100/011000/100/100000/010/100000/011/100000/011/010000/011/011000/001/010000/100/011000/100/100000/100/000000/000/000000/000/000000/000/000000/000



The Secret Key
(Removing "/")

100000001100000001100000100100000010011000001011000100011000100011000100100000010100000011100000011010000011011000001010000100011000100100000100000000000000000000000000000000000000000

180-bit Secret Key

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.

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

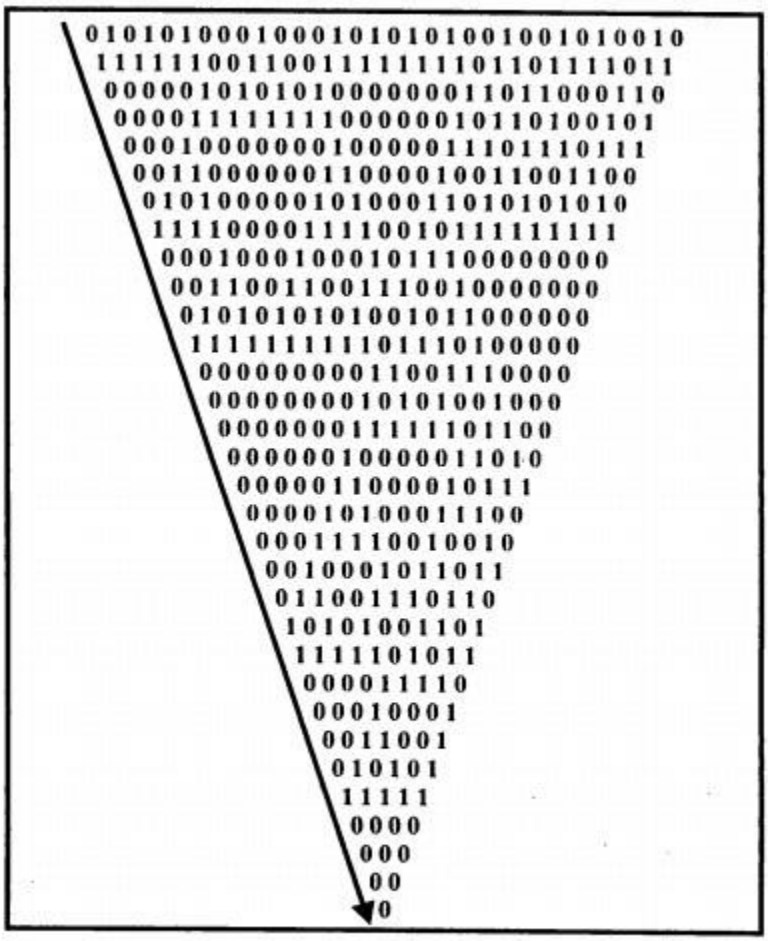
"/" being working as the separator between two consecutive bytes.

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

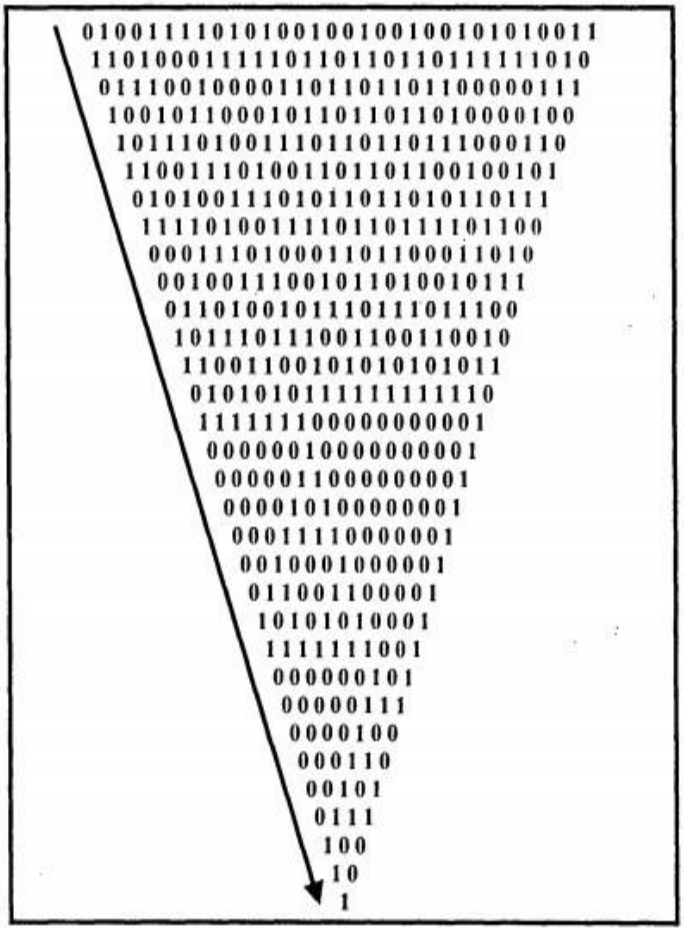
Different Blocks Considered for Encryption

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

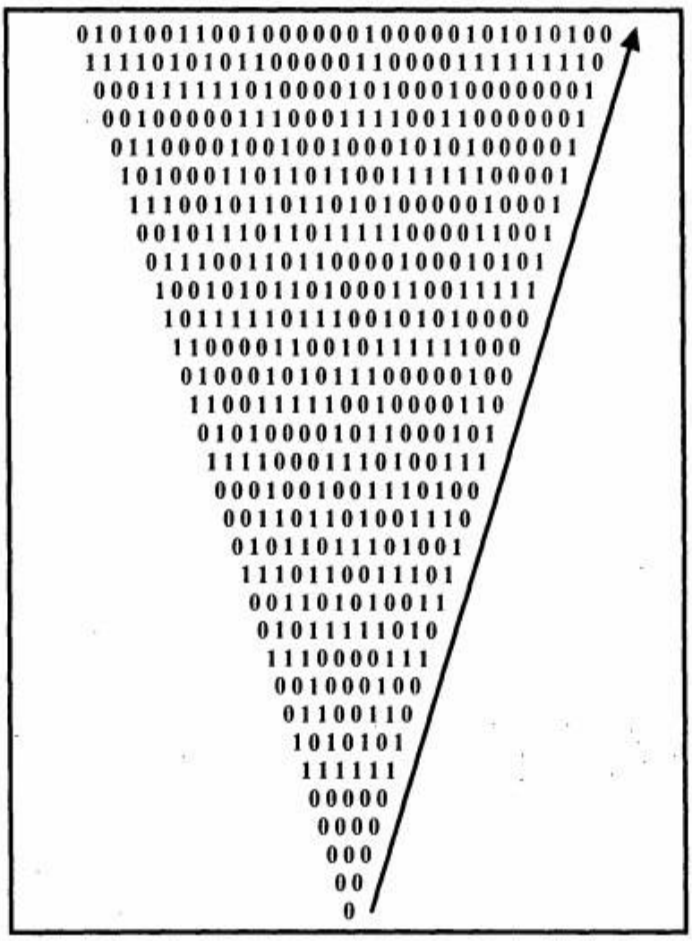
the formations of all the triangles corresponding to block 1 to block 16 respectively. Along with the triangle, each figure also shows the corresponding target block following the option mentioned against each block in table



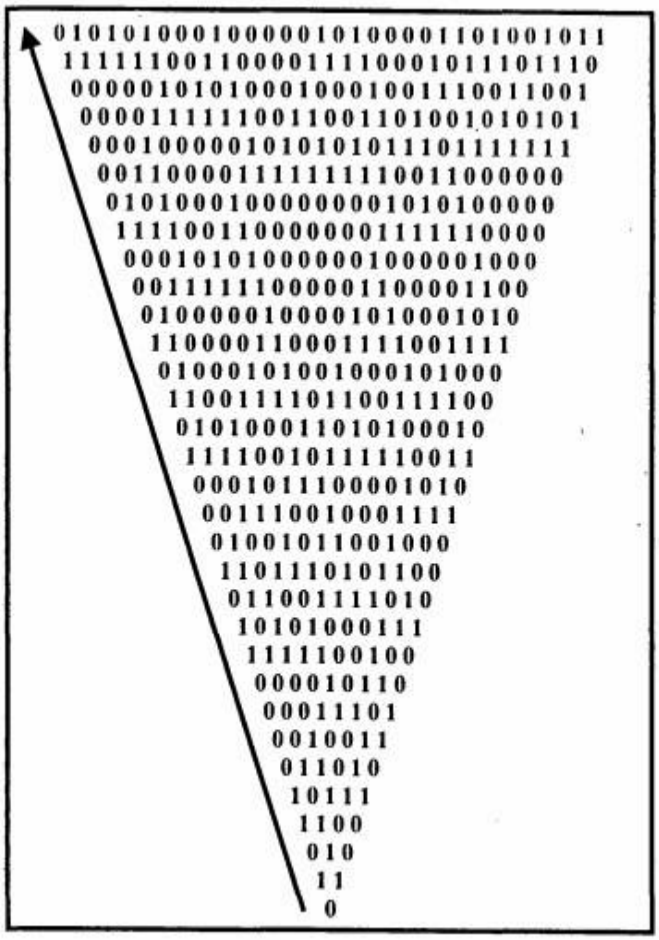
Construction of Target Block for Block 1 from the Generated Triangle



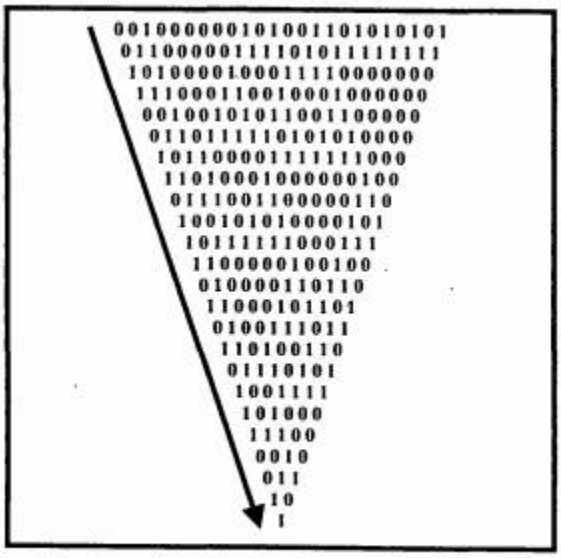
Construction of Target Block for Block 2 from the Generated Triangle



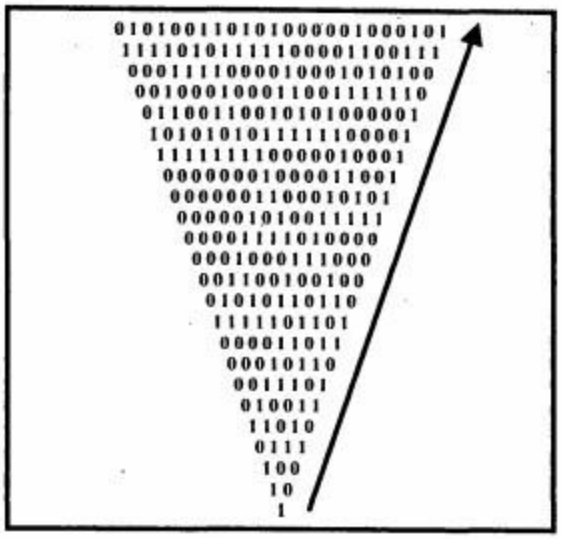
Construction of Target Block for Block 3 from the Generated Triangle



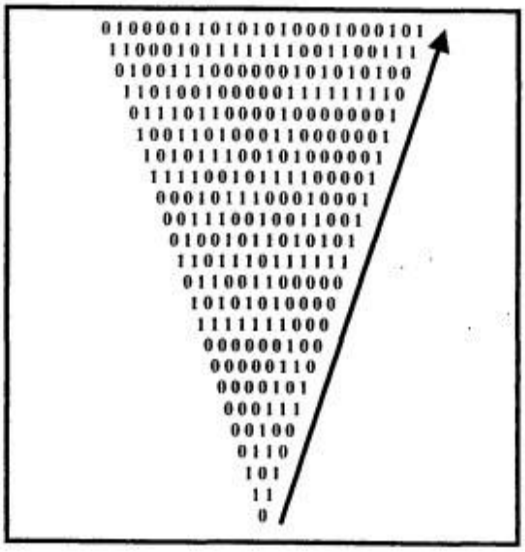
Construction of Target Block for Block 4 from the Generated Triangle



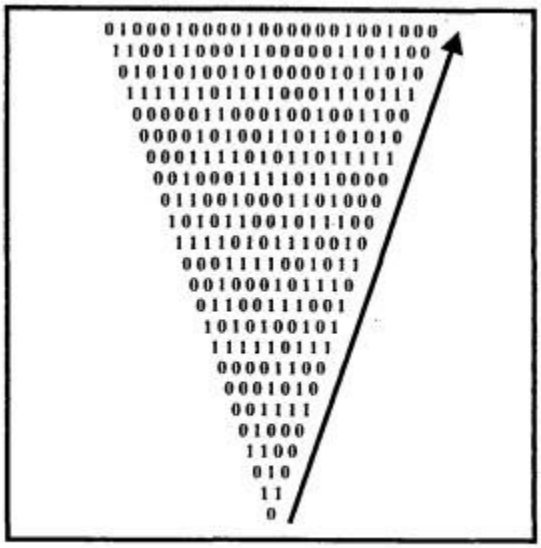
Construction of Target Block for Block 5 from the Generated Triangle



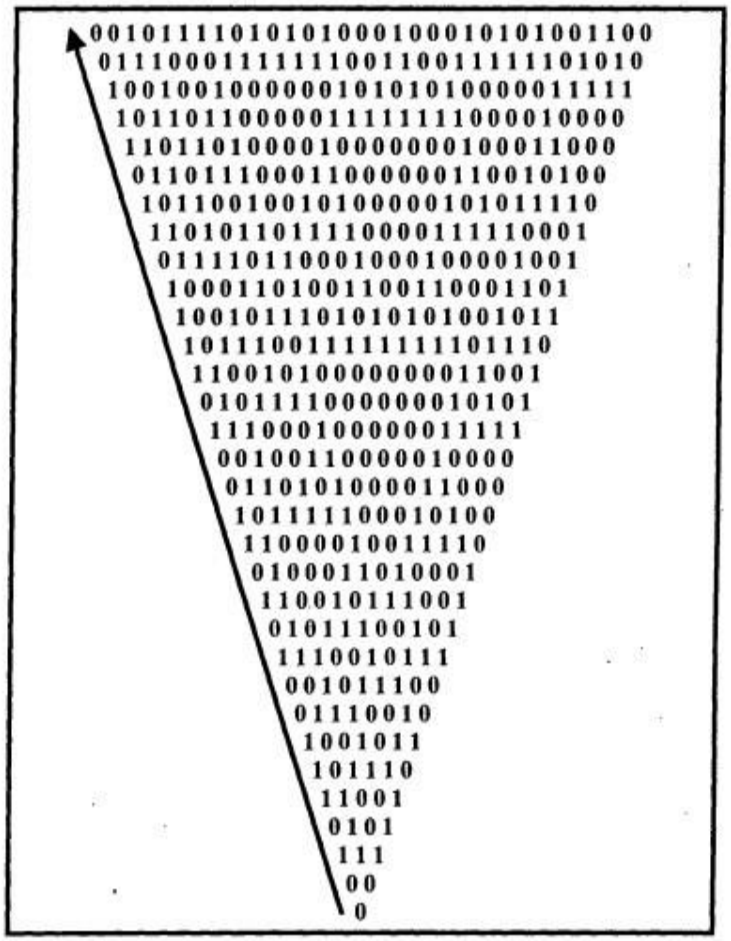
Construction of Target Block for Block 6 from the Generated Triangle



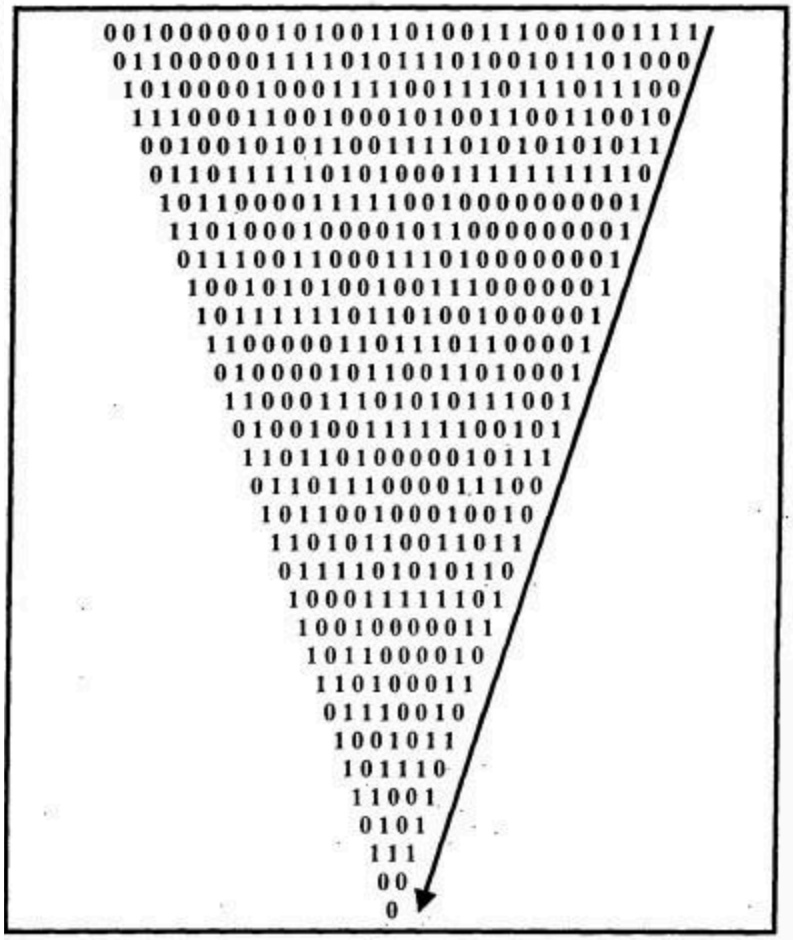
Construction of Target Block for Block 7 from the Generated Triangle



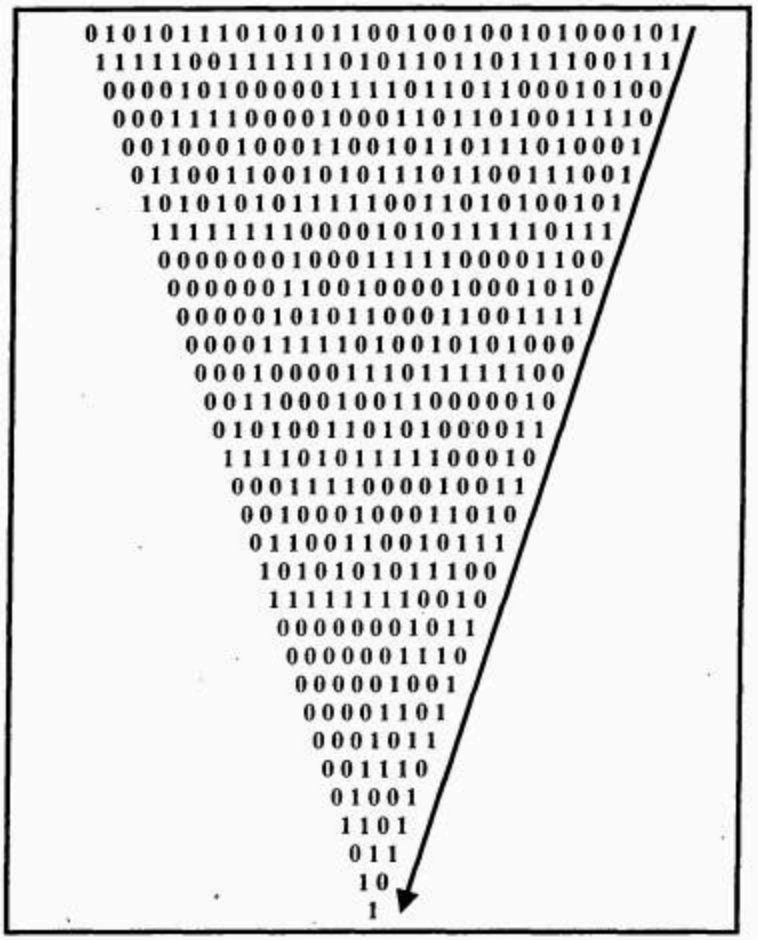
Construction of Target Block for Block 8 from the Generated Triangle



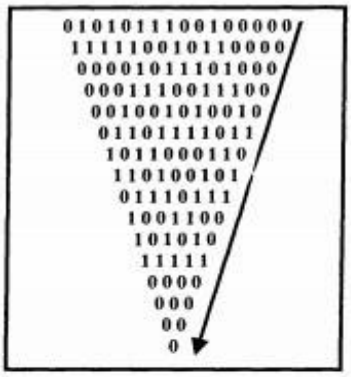
Construction of Target Block for Block 9 from the Generated Triangle



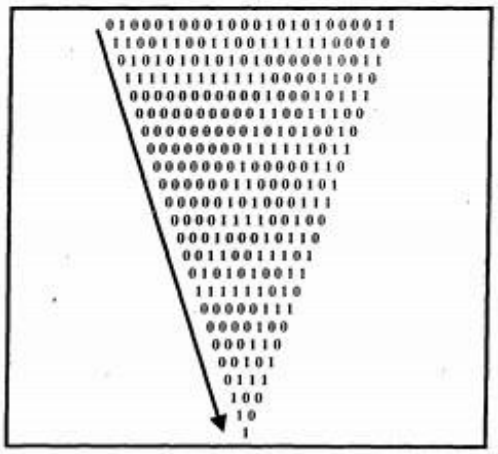
Construction of Target Block for Block 10 from the Generated Triangle



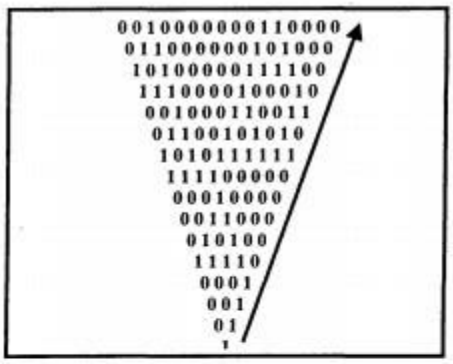
Construction of Target Block for Block 11 from the Generated Triangle



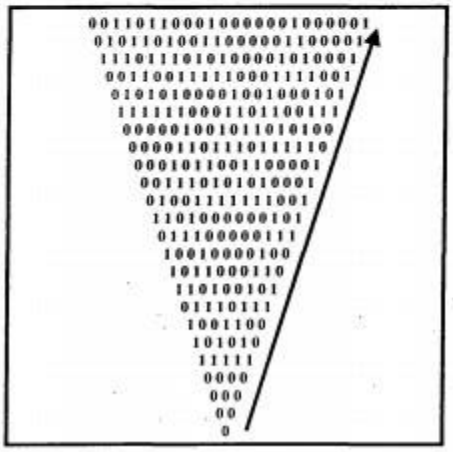
Construction of Target Block for Block 12 from the Generated Triangle



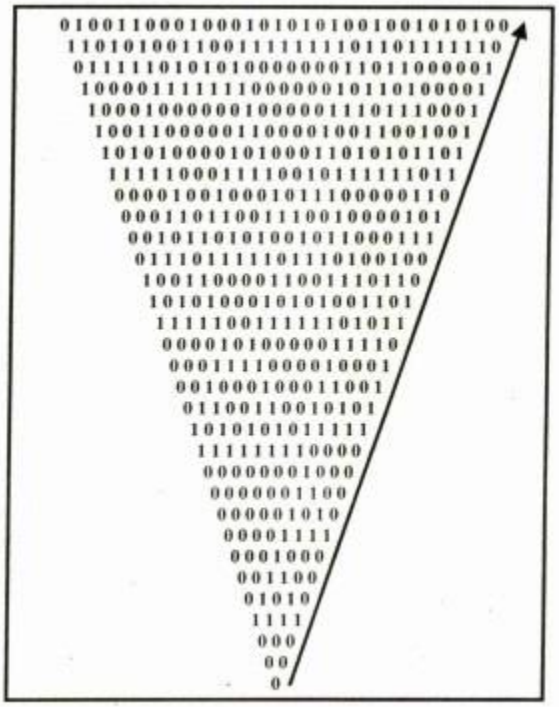
Construction of Target Block for Block 13 from the Generated Triangle



Construction of Target Block for Block 14 from the Generated Triangle



Construction of Target Block for Block 15 from the Generated Triangle



Construction of Target Block for Block 16 from the Generated Triangle

All Target Blocks

Target Block 1	**01000001000100000000011000010000**
Target Block 2	**01011101000110100000011000000111**
Target Block 3	**00000110010111001100001111111100**
Target Block 4	**01011000011010001010100010000010**
Target Block 5	**00110011011101010110011**
Target Block 6	**10010110110000111111110011**
Target Block 7	**01100110000011111111110011**
Target Block 8	**01000100111010000101001000**
Target Block 9	**00101110010101100101111011011100**
Target Block 10	**10001011111111111001011010101011100**
Target Block 11	**11001111001000101010010111011101**
Target Block 12	**0000010110010000**
Target Block 13	**01010000000000000100000111**
Target Block 14	**11110000010100000**
Target Block 15	**00001001100111111001111111**
Target Block 16	**00010001000011110110011011111100**

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

01000001000100000000011000010000010111010001101000000011000000011100000110
01011100110000111111111000101100000110100010101000100000010001100110111010101
01110011100101101100001111111001101100110000001111111110011010000100111010000
01001000001011100101011001011111011011100100010111111111110010110101011100
11001111001000101010010111011101000001011001000001010000000000001000001111
11110000010100000000010011001111100111111100010001000011110110011011111100

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

01000001/00010000/00000110/00010000/01011101/00011010/00000110/00000111/000
00110/01011100/11000011/11111100/01011000/01101000/10101000/10000010/001100
11/01110101/01110011/10010110/11000011/11110011/01100110/00001111/11110011/
01000100/11101000/01001000/00101110/01010110/01011110/11011100/10001011/111
11111/00101101/01011100/11001111/00100010/10100101/11011101/00000101/100100
00/01010000/00000001/00000111/11110000/01010000/00001001/10011111/00111111/
00010001/00001111/01100110/11111100

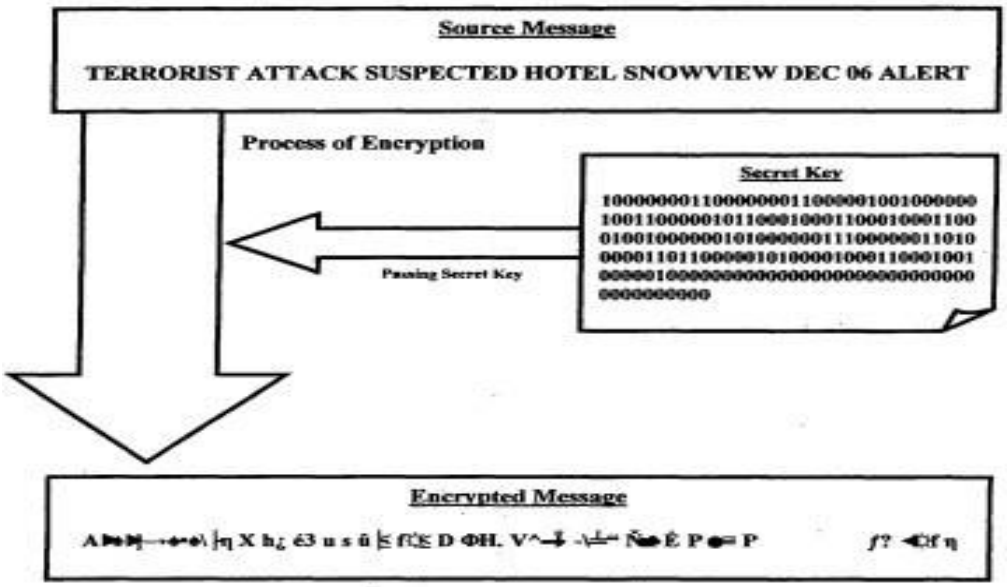
Converting Bytes into Characters

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

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

"A►►¶→←←↕\ ├η X h¿ é3 u s û ≤ f☼≤ D ΦH. V^▄↓ ┴╧" Ñ► É P ●═ P ƒ? ◄☼f η"

This is the ciphertext to be transmitted corresponding to the plaintext taken for the implementation purpose.



Generating Encrypted Message from Source using 180-Bit Secret Key

4. Chi square Calculation :

Through the chi square test performed between the original and the encrypted files, the non-homogeneity of the two files is tested. The "Pearsonian Chi-square test" or the "Goodness-of-fit Chi-square test" has been performed here to decide whether the observations onto encrypted files arc in good agreement with a hypothetical distribution, which means whether the sample of encrypted files may be supposed to have arisen from a specified population. In this case, the chi square distribution is being performed with (2-1) = 1 degree of freedom, 2 being the total number of classes of possible characters in the source as well as in the encrypted files. If the observed value of the statistic exceeds the tabulated value at a given level, the null hypothesis is rejected. The "Pearsonian Chi-square" or the "Goodness-of-fit Chi square" is defined as follows:

$$X^2 = \sum\{(f_0 - f_c)^2 / f_c\}$$

Here fc and fO respectively stand for the frequencies of '0' and 'I' in the source file and that of the same character in the corresponding encrypted file. On the basis of this formula, the Chi-square values have been calculated for sample pairs of source and encrypted files.

5. Results :

For the purpose of implementation a unique size of 64 bit has been considered, and for each such block, a symmetric technique has been implemented. In the report below the section represents the Encryption/Decryption times and Chi square values. Also section 5.1 shows the result on .CPP files, section 5.2 shows the result on .SYS files, section 5.3 shows the result on .TXT files, section 5.4 shows the result on .DLL files, and section 5.5 shows the result on .EXE files.

5.1.1 Results Of .CPP files :

Table 5.1.1.1 gives the result of implementing the technique on .CPP files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 0.714286 seconds to 0.384615 seconds. The decryption time ranges from 0.769231 seconds to 0.43956 seconds. The Chi Square value ranges from 68.626 to 17.336 and the degree of freedom ranges from 70 to 90.

Table 5.1.1.1

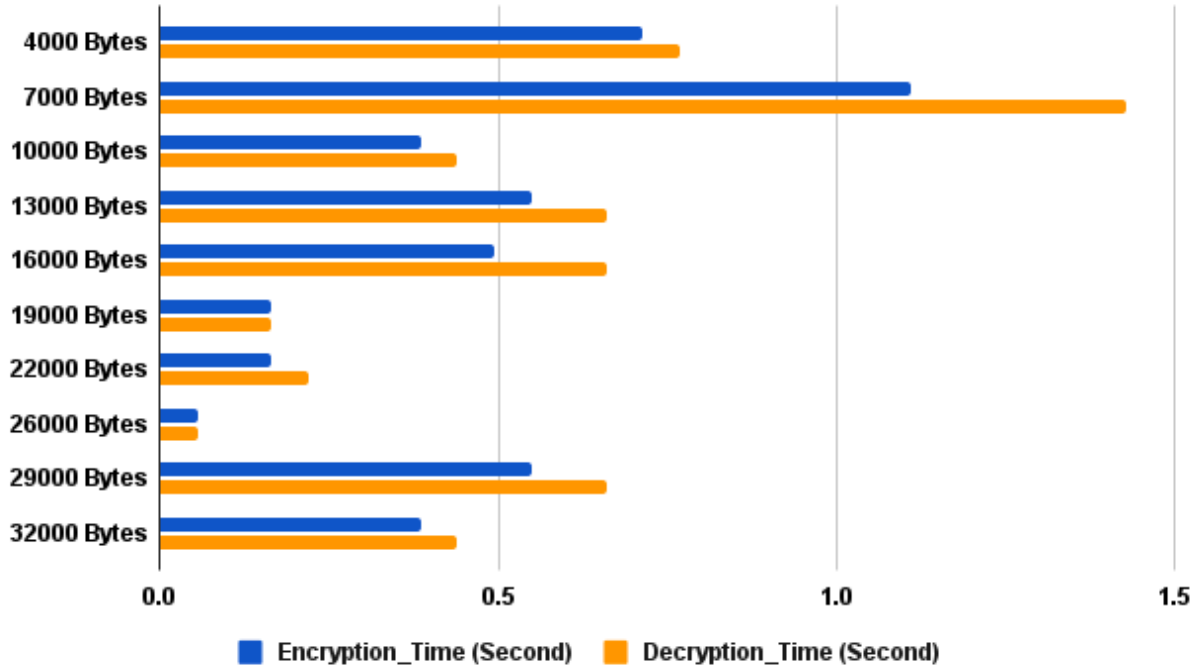
Result for .CPP files for TE technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	0.714286	0.769231	68.626	88
test_file_02	7000 Bytes	1.108791	1.428571	144.422	90
test_file_03	10000 Bytes	0.384615	0.43956	15.466	77
test_file_04	13000 Bytes	0.549451	0.659341	47.550	88
test_file_05	16000 Bytes	0.494505	0.659341	45.205	84
test_file_06	19000 Bytes	0.164835	0.164835	83.87	70
test_file_07	22000 Bytes	0.164835	0.21978	79.16	76
test_file_08	26000 Bytes	0.054945	0.054945	13.05	83
test_file_09	29000 Bytes	0.549451	0.659341	48.061	88
test_file_10	32000 Bytes	0.384615	0.43956	17.336	74

A part of the table diagrammatically represents in figure 5.1.1.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .CPP file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 5.1.1.2

Comparison of encryption and decryption time of .CPP files



5.1.2. Results Of .SYS files :

Table 5.1.2.1 gives the result of implementing the technique on .SYS files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 1.263736 seconds to 1.153846 seconds. The decryption time ranges from 1.428571 seconds to 1.373626 seconds. The Chi Square value ranges from 63.553 to 12.6367 and the degree of freedom ranges from 240 to 255.

Table 5.1.2.1

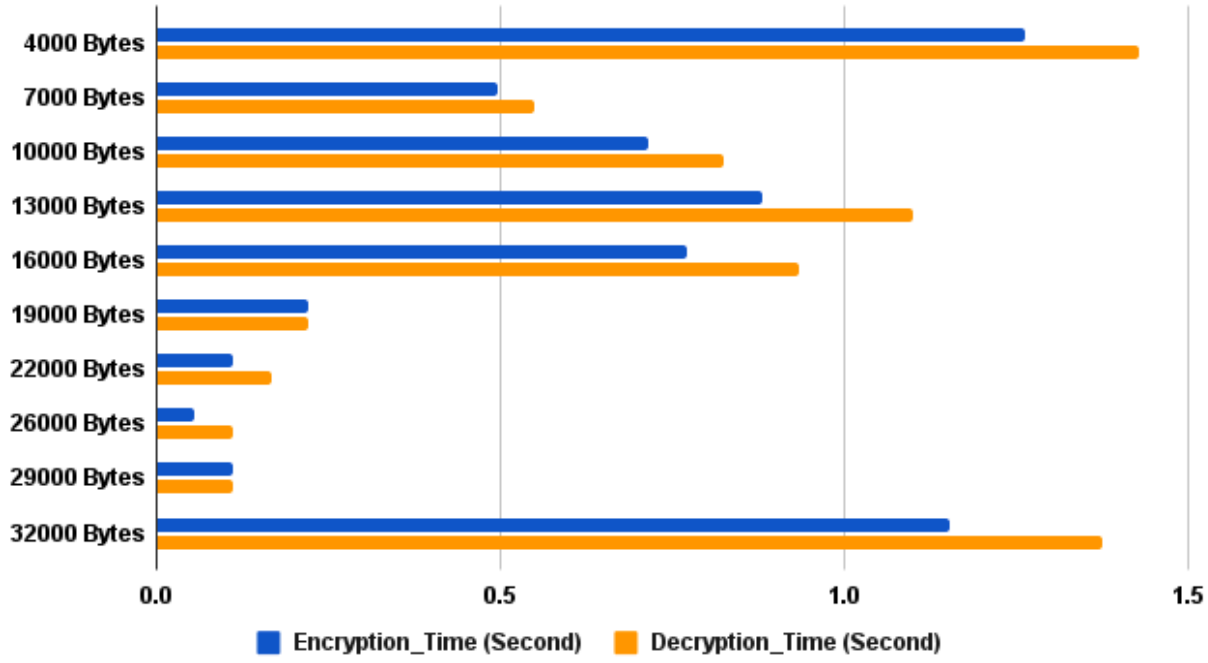
Result for .SYS files for TE technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	1.263736	1.428571	63.553	255
test_file_02	7000 Bytes	0.494505	0.549451	16.452	241
test_file_03	10000 Bytes	0.714286	0.824176	84.593	255
test_file_04	13000 Bytes	0.879121	1.098901	65.706	255
test_file_05	16000 Bytes	0.769231	0.934066	58.201	255
test_file_06	19000 Bytes	0.21978	0.21978	21.052	250
test_file_07	22000 Bytes	0.10989	0.164835	64.88	255
test_file_08	26000 Bytes	0.054945	0.10989	8.59	240
test_file_09	29000 Bytes	0.10989	0.10989	3.19	255
test_file_10	32000 Bytes	1.153846	1.373626	12.6367	255

A part of the table diagrammatically represents in figure 5.1.2.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .SYS file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 5.1.2.2

Comparison of encryption and decryption time of .SYS files



5.1.3. Results Of .TXT files :

Table 5.1.3.1 gives the result of implementing the technique on .TXT files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 0.769231 seconds to 0.714286 seconds. The decryption time ranges from 0.934066 seconds to 0.879121 seconds. The Chi Square value ranges from 33.211 to 43.706 and the degree of freedom ranges from 220 to 255.

Table 5.1.3.1

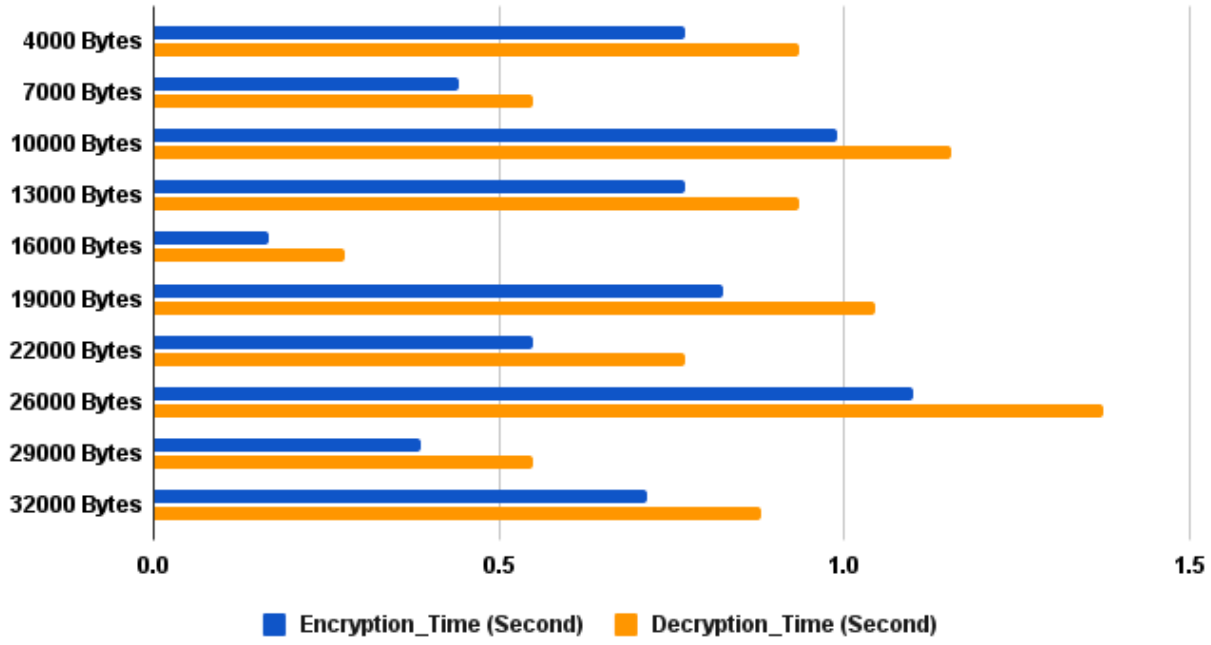
Result for .TXT files for TE technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	0.769231	0.934066	33.211	255
test_file_02	7000 Bytes	0.43956	0.549451	28.716	250
test_file_03	10000 Bytes	0.989011	1.153846	47.792	254
test_file_04	13000 Bytes	0.769231	0.934066	48.939	232
test_file_05	16000 Bytes	0.164835	0.274725	9.254	240
test_file_06	19000 Bytes	0.824176	1.043956	51.815	234
test_file_07	22000 Bytes	0.549451	0.769231	39.119	255
test_file_08	26000 Bytes	1.098901	1.373626	63.314	255
test_file_09	29000 Bytes	0.384615	0.549451	6.451	220
test_file_10	32000 Bytes	0.714286	0.879121	43.706	255

A part of the table diagrammatically represents in figure 5.1.3.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .TXT file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 5.1.3.2

Comparison of encryption and decryption time of .TXT files



5.1.4 Results Of .DLL files :

Table 5.1.4 gives the result of implementing the technique on .DLL files. Ten files have been considered. Their sizes range from 4000 bytes to 5000 bytes. The encryption time ranges from 1.208791 seconds to 1.153846 seconds. The decryption time ranges from 1.483516 seconds to 1.428571 seconds. The Chi Square value ranges from 70.479 to 205.511 and the degree of freedom ranges from 215 to 255.

Table 5.1.4.1

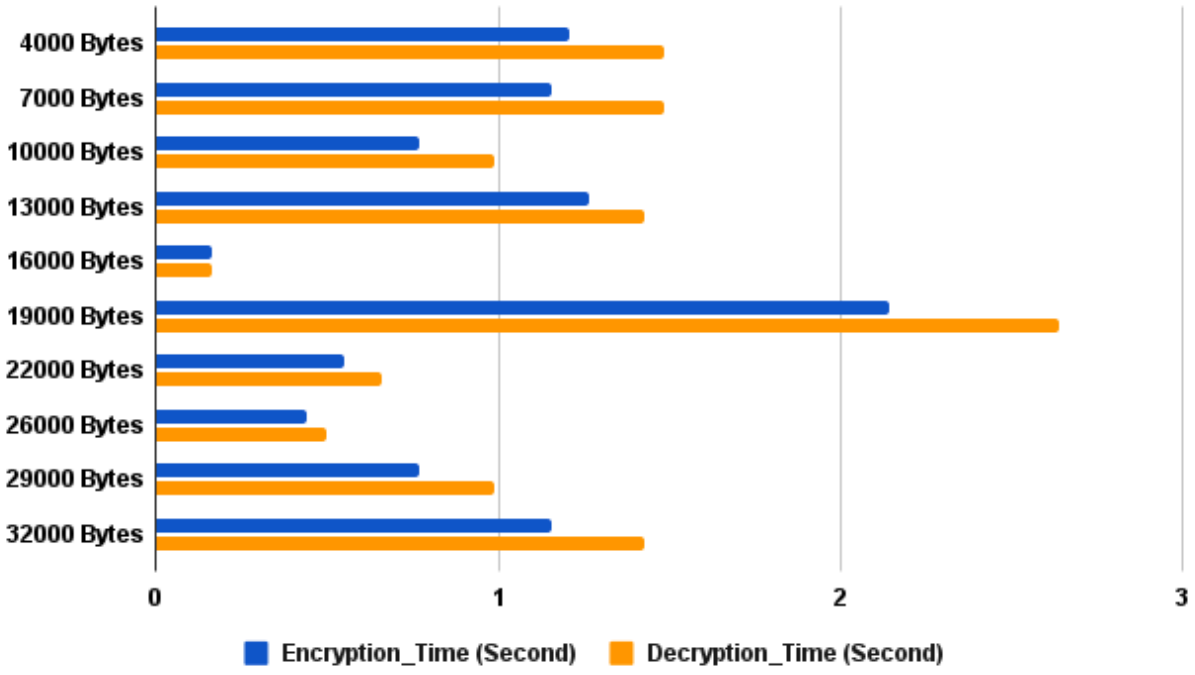
Result for .DLL files for TE technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	1.208791	1.483516	70.479	253
test_file_02	7000 Bytes	1.153846	1.483516	20.3671	254
test_file_03	10000 Bytes	0.769231	0.989011	10.5418	252
test_file_04	13000 Bytes	1.263736	1.428571	18.2841	217
test_file_05	16000 Bytes	0.164835	0.164835	83.83	255
test_file_06	19000 Bytes	2.142857	2.637362	153.199	250
test_file_07	22000 Bytes	0.549451	0.659341	56.969	255
test_file_08	26000 Bytes	0.43956	0.494505	60.992	255
test_file_09	29000 Bytes	0.769231	0.989011	168.116	215
test_file_10	32000 Bytes	1.153846	1.428571	205.511	252

A part of the table diagrammatically represents in figure 5.1.4.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .DLL file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 5.1.4.2

Comparison of encryption and decryption time of .DLL files



5.1.5. Results Of .EXE files :

Table 5.1.5.1 gives the result of implementing the technique on .EXE files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 1.428571 seconds to 0.714286 seconds. The decryption time ranges from 1.703297 seconds to 0.879121 seconds. The Chi Square value ranges from 128.674 to 158.628 and the degree of freedom ranges from 217 to 255.

Table 5.1.5.1

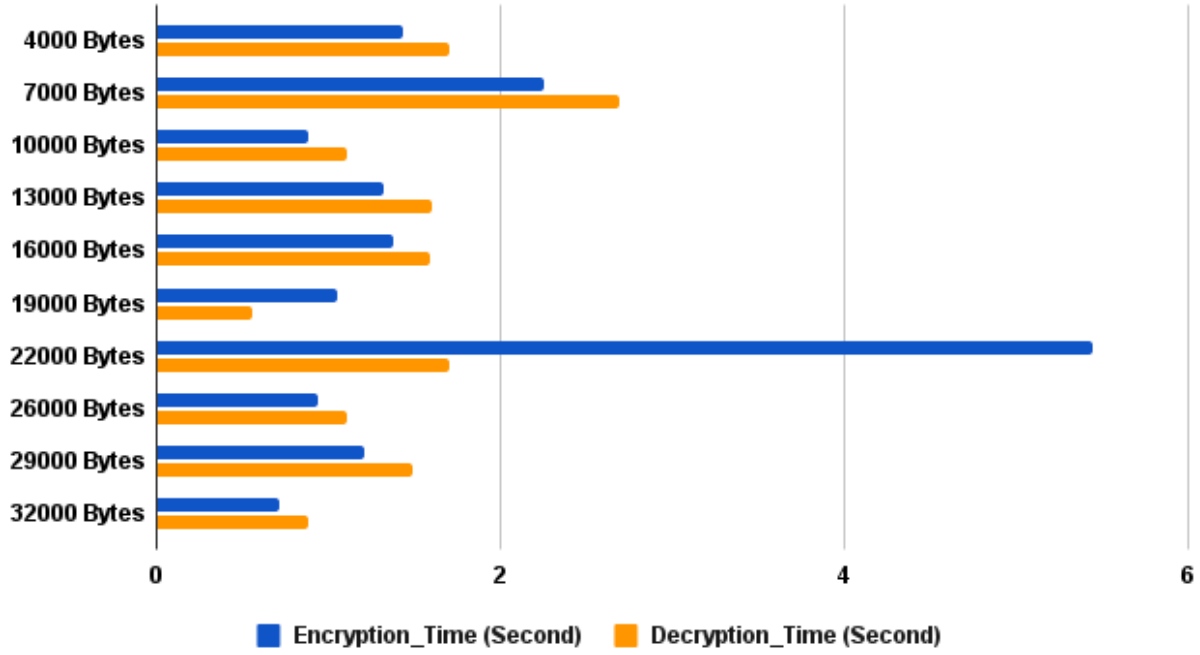
Result for .EXE files for TE technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	1.428571	1.703297	128.674	255
test_file_02	7000 Bytes	2.252747	2.692307	158.628	255
test_file_03	10000 Bytes	0.879121	1.098901	17.993	254
test_file_04	13000 Bytes	1.318681	1.593407	55.351	250
test_file_05	16000 Bytes	1.373626	1.58242	62.986	217
test_file_06	19000 Bytes	1.043956	0.549451	33.539	248
test_file_07	22000 Bytes	5.43956	1.703297	57.961	220
test_file_08	26000 Bytes	0.934066	1.098901	79.747	253
test_file_09	29000 Bytes	1.208791	1.483516	110.332	255
test_file_10	32000 Bytes	0.714286	0.879121	65.56	255

A part of the table diagrammatically represents in figure 5.1.5.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .EXE file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 5.1.1.2

Comparison of encryption and decryption time of .EXE files



6. TE with Session Key

In this section we are implementing the TE algorithm with a session key and then implement the code with the previous files. The session key is 128 bit long with 8 segment assignments of 16 bits, 16 bits, 16 bits, 16 bits, 16 bits, 16 bits, 16 bits, 16 bits. Details on the making of the session key is given below at section 6.1. Then the file is divided according to the session key. Details on the implementation are given in section 6.2. Finally we test the result of the code and the results are declared in section 7.

6.1. Generating the Session Key

A 128 bits session key is proposed with 8 segment of size 16 bits

16 bits	16 bits	16 bits	16 bits	16 bits	16 bits	16 bits	16 bits

According to the formation of the session key we are dividing the input file. The dividing logic is detailed in the next section.

6.2. File Dividing Logic :

A session key is a temporary encryption key used to secure a single communication session. It is generated for a specific session and discarded after the session ends. This key is used for encryption and decryption of data during the session, ensuring the confidentiality and integrity of the transmitted data. Session keys are commonly used in cryptography protocols to provide secure communication over the network.

This method ensures that each session has a unique encryption key, enhancing the security by preventing the reuse of keys across different sessions.

- **Example of key - Generation - 128 bit key :**

Block_size	Operation
0000000000100010 (16 bits)(34 blocks/34 bits)	**010 (3 Blocks / 3 bits)**
0000000001001100 (16 bits)(76 blocks/76 bits)	**100 (3 Blocks / 3 bits)**
0000000001111010 (16 bits)(58 blocks/58 bits)	**001 (3 Blocks / 3 bits)**
0000000001011011 (16 bits)(91 blocks/91 bits)	**011 (3 Blocks / 3 bits)**
0000000001010010 (16 bits)(82 blocks/82 bits)	**010 (3 Blocks / 3 bits)**
0000000001000011 (16 bits)(67 blocks/67 bits)	**001 (3 Blocks / 3 bits)**
0000000001101111 (16 bits)(55 blocks/55 bits)	**100 (3 Blocks / 3 bits)**
0000000001100010 (16 bits)(49 blocks/49 bits)	**011 (3 Blocks / 3 bits)**

- **Total** : 34+2+76+4+58+1+91+3+82+2+67+1+55+4+49+3 = 532 Blocks

- **Size of the file for the session key :**
  34+2+76+4+58+1+91+3+82+2+67+1+55+4+49+3 = 532 Blocks
- (34×34)+(2×2)+(76×76)+(4×4)+(58×58)+(1×1)+(91×91)+(3×3)+(82×82)+(2×2)+(67×67)+(1×1)+(55×55)+(4×4)+(49×49)+(3×3)
  = 28376 + 4 = 28380 bits = 3547 bytes

The above calculation describes the Triangulation Encryption (TE) technique with a 128 bits key session key, divided into blocks of varying sizes (34 to 91 bits)and the operations (3 bits each). The total number of blocks is 532, and the session key size is 28380 bits (3547 bytes). Using this session key provides more protection compared to a normal secret key.

6.3. Implementation :

Each part of the content is converted into binary format. The length of each binary 0 of chunk is determined by the shuffled bit lengths. For example, if the bit length is 8, each chunk is one character long; if it's 16, each chunk is two characters long, and so on.

The binary chunks from each part of the content are stored in a new Iist.This new list holds the binary representation of each content corresponding to the shifted bit length.

The binary chunks are then mapped to their respective bit lengths using a dictionary. This dictionary associates each bit length with the corresponding list of binary chunks.

shuffled list : [16, 16, 16, 16, 16, 16, 16, 16]

The process goes through each key-value pair in the dictionary that maps bit kegs to lists of binary chunks. :After this the implementation is the same as the implementation of TE algorithm which was mentioned before.

7. Results of TE with Session Key :

The TE with session key is tested on different file formats with varying file sizes. The results of .txt files are in section 7.1, .sys files are in section 7.2, .cpp files are in ration 7.3, .dll files are in section 7.4, and .exe files are in section 7.5.

This testing the files that were used for testing the TE algorithms are used to test TE with Session Key. There is no change of files in both testing. The session key is 128 bit long with 8 segments. Consisting of 16 bits each. The bit division of session key every segment can store minimum 0 to maximum 2^{16} -1 bit of block and each block can can have minimum 0 to maximum 2^{16} -1 bits of data.

While testing this algorithm the encryption time, decryption time varied greatly, chi-square value and degree of freedom sometimes varied greatly with no testing as the session key greatly differs each time. The number of xor operations while encrypting and decrypting are the only values that were similar in every test. So to keep consistent I have tested every file five times and all the results are the average of the five times testing.

7.1. Results on .TXT Files :

Table 7.1.1 gives the result of implementing the technique on .TXT files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 0.156099 seconds to 17.625914 seconds. The decryption time ranges from 0.156099 seconds to 17.625914 seconds. The Chi Square value ranges from 128.587797 to 470.999592 and the degree of freedom ranges from 230 to 255.

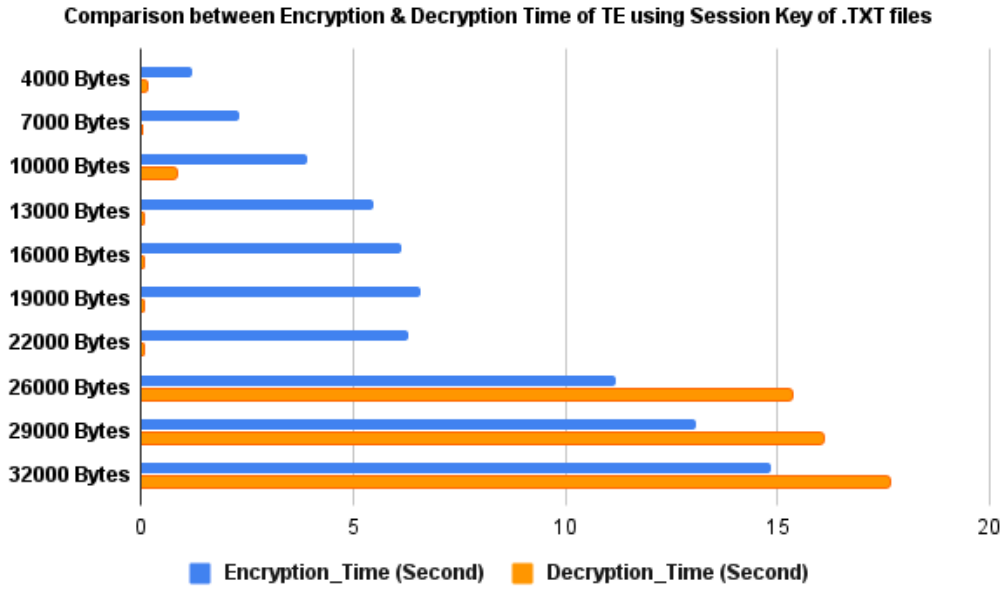
Table 7.1.1

Results For .TXT Files For TE With Session Key Technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	1.202979	0.156099	128.587797	255
test_file_02	7000 Bytes	2.317999	0.031128	70.616949	254
test_file_03	10000 Bytes	3.903669	0.828124	148.730185	230
test_file_04	13000 Bytes	5.500979	0.078132	420.281077	242
test_file_05	16000 Bytes	6.12619	0.078123	482.817114	235
test_file_06	19000 Bytes	6.58763	0.045716	399.414111	233
test_file_07	22000 Bytes	6.293681	0.078094	366.628815	255
test_file_08	26000 Bytes	11.187337	15.35204	341.488989	255
test_file_09	29000 Bytes	13.062765	16.0625	380.896572	232
test_file_10	32000 Bytes	14.846441	17.625914	470.999592	230

A part of the table diagrammatically represents in figure 7.1.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .TXT file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 7.1.2



7.2. Results on .EXE Files :

Table 7.2.1 gives the result of implementing the technique on .EXE files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 0.171914 seconds to 1.553301 seconds. The decryption time ranges from 1.718597 seconds to 20.753486 seconds. The Chi Square value ranges from 51.782203 to 375.870778 and the degree of freedom ranges from 220 to 255.

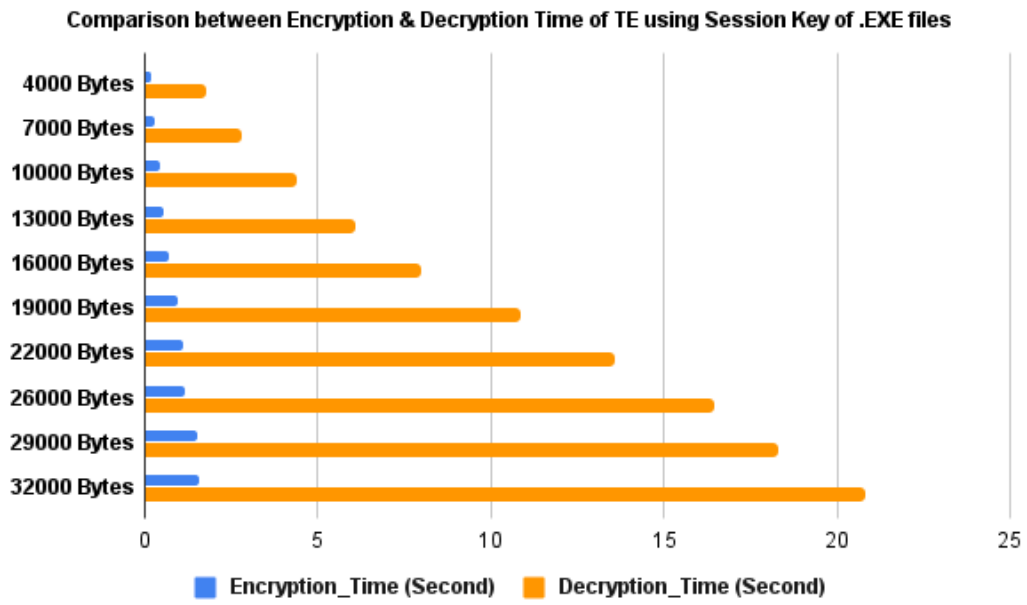
Table 7.2.1

Results For .EXE Files For TE With Session Key Technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	0.171914	1.718597	51.782203	231
test_file_02	7000 Bytes	0.281152	2.749832	270.105346	220
test_file_03	10000 Bytes	0.437495	4.328413	191.43711	255
test_file_04	13000 Bytes	0.531099	6.032951	435.293537	255
test_file_05	16000 Bytes	0.687533	7.908084	112.924784	255
test_file_06	19000 Bytes	0.937646	10.782696	568.219856	255
test_file_07	22000 Bytes	1.095379	13.502474	346.913628	248
test_file_08	26000 Bytes	1.156127	16.409009	447.017548	255
test_file_09	29000 Bytes	1.514812	18.254904	395.195554	249
test_file_10	32000 Bytes	1.553301	20.753486	375.870778	255

A part of the table diagrammatically represents in figure 7.2.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .EXE file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 7.2.2



7.3. Results on .CPP Files :

Table 7.3.1 gives the result of implementing the technique on .CPP files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 0.124855 seconds to 1.624865 seconds. The decryption time ranges from 1.844999 seconds to 22.111212 seconds. The Chi Square value ranges from 70.413359 to 420.889532 and the degree of freedom ranges from 66 to 90.

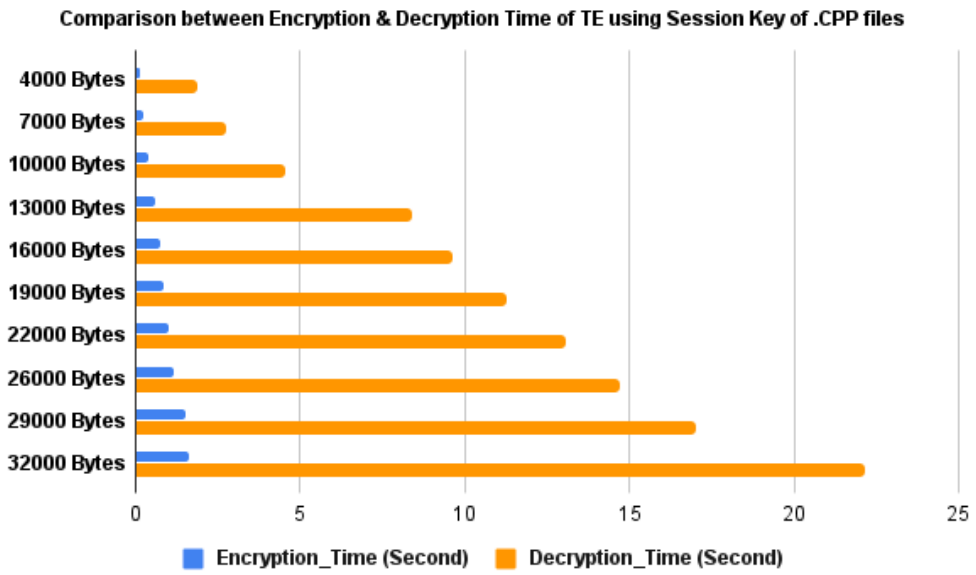
Table 7.3.1

Results For .CPP Files For TE With Session Key Technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	0.124855	1.844999	70.413359	88
test_file_02	7000 Bytes	0.249854	2.68735	56.182209	90
test_file_03	10000 Bytes	0.390523	4.486903	111.079143	77
test_file_04	13000 Bytes	0.609253	8.331201	42.86637	84
test_file_05	16000 Bytes	0.718598	9.561033	378.055316	89
test_file_06	19000 Bytes	0.843714	11.236791	301.934444	83
test_file_07	22000 Bytes	1.01566	13.033091	299.544018	69
test_file_08	26000 Bytes	1.168467	14.639646	403.029631	88
test_file_09	29000 Bytes	1.500023	16.971841	330.719072	73
test_file_10	32000 Bytes	1.624865	22.111212	420.889532	66

A part of the table diagrammatically represents in figure 7.3.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .CPP file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 7.3.2



7.4. Results on .SYS Files :

Table 7.4.1 gives the result of implementing the technique on .SYS files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 0.124831 seconds to 1.57794 seconds. The decryption time ranges from 1.752299 seconds to 21.8616 seconds. The Chi Square value ranges from 72.057296 to 377.392064 and the degree of freedom ranges from 215 to 255.

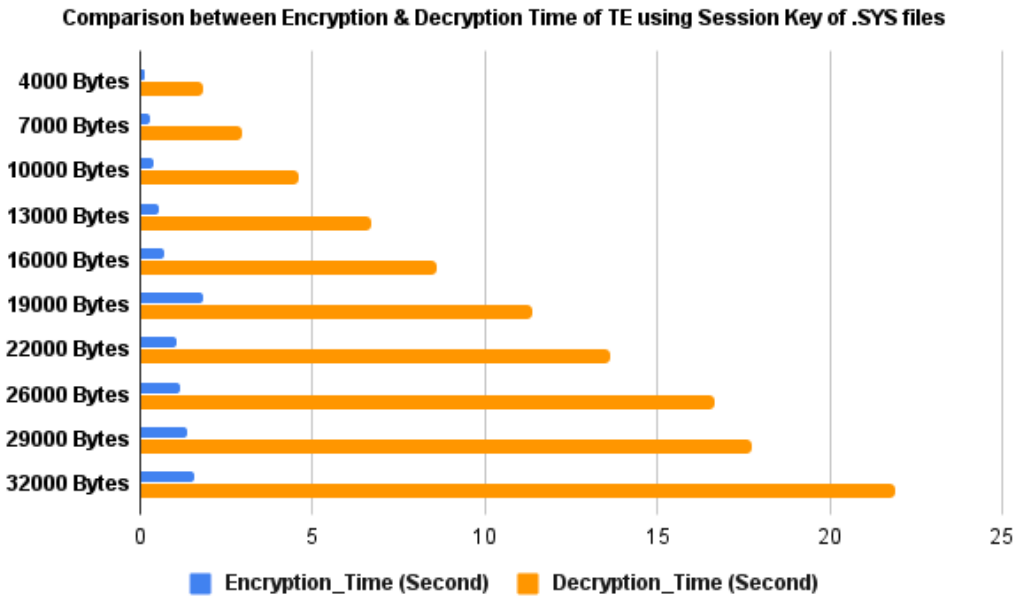
Table 7.4.1

Results For .SYS Files For TE With Session Key Technique

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	0.124831	1.752299	72.057296	241
test_file_02	7000 Bytes	0.298541	2.896509	80.180298	255
test_file_03	10000 Bytes	0.390962	4.549235	125.184365	255
test_file_04	13000 Bytes	0.531235	6.62544	244.64807	250
test_file_05	16000 Bytes	0.687729	8.563719	50.819086	237
test_file_06	19000 Bytes	1.823219	11.311034	202.444999	265
test_file_07	22000 Bytes	1.031333	13.564215	339.475828	215
test_file_08	26000 Bytes	1.140479	16.594119	446.223894	255
test_file_09	29000 Bytes	1.343932	17.69602	245.493039	255
test_file_10	32000 Bytes	1.57794	21.8616	377.392064	255

A part of the table diagrammatically represents in figure 7.4.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .SYS file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 7.4.2



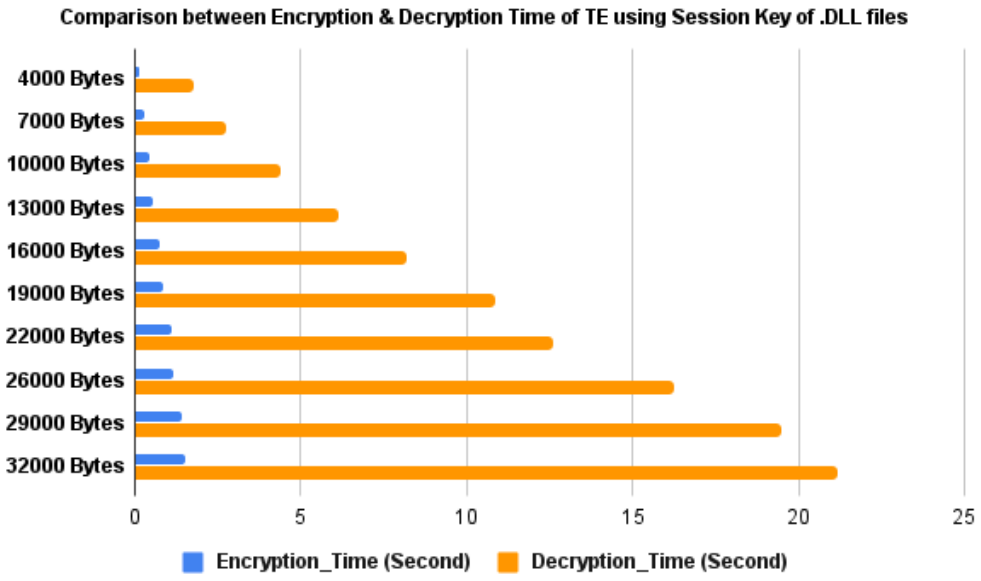
7.5. Results on .DLL Files :

Table 7.5.1 gives the result of implementing the technique on .DLL files. Ten files have been considered. Their sizes range from 4000 bytes to 32000 bytes. The encryption time ranges from 0.140458 seconds to 1.515615 seconds. The decryption time ranges from 1.719002 seconds to 21.130331 seconds. The Chi Square value ranges from 55.807867 to 541.236572 and the degree of freedom ranges from 217 to 255.

Source File	Size(In Bytes)	Encryption Time(Second)	Decryption Time(Second)	Chi-Square Value	Degree Of Freedom
test_file_01	4000 Bytes	0.140458	1.719002	55.807867	253
test_file_02	7000 Bytes	0.296928	2.710667	139.893625	254
test_file_03	10000 Bytes	0.453143	4.313909	262.679266	252
test_file_04	13000 Bytes	0.544144	6.063477	573.974581	255
test_file_05	16000 Bytes	0.750039	8.142087	220.569941	217
test_file_06	19000 Bytes	0.843706	10.828967	95.466473	255
test_file_07	22000 Bytes	1.09358	12.532599	174.715432	250
test_file_08	26000 Bytes	1.171974	16.205546	280.932442	251
test_file_09	29000 Bytes	1.4065	19.408746	433.015168	247
test_file_10	32000 Bytes	1.515615	21.130331	541.236572	255

A part of the table diagrammatically represents in figure 7.5.2, where one graphical relationship is established between the source size and the encryption time and decryption time for the .DLL file. From the figure, it can be interpreted that there is a tendency that the encryption time changes almost linearly with the size of the source file.

Table 7.5.2



8. Comparison between Encryption and Decryption Time of TE and TE using Session Key

In this section we will compare the encryption time decryption time of various files using the TE algorithm with the files using the TE algorithm with session key.

8.1.1. Comparison between Encryption Time of TE and TE using Session Key of .TXT files

8.1.2. Comparison between Decryption Time of TE and TE using Session Key of .TXT files

8.2.1. Comparison between Encryption Time of TE and TE using Session Key of .EXE files

8.2.2. Comparison between Decryption Time of TE and TE using Session Key of .EXE files

8.3.1. Comparison between Encryption Time of TE and TE using Session Key of .CPP files

8.3.2. Comparison between Decryption Time of TE and TE using Session Key of .CPP files

8.4.1. Comparison between Encryption Time of TE and TE using Session Key of .SYS files

8.4.2. Comparison between Decryption Time of TE and TE using Session Key of .SYS files

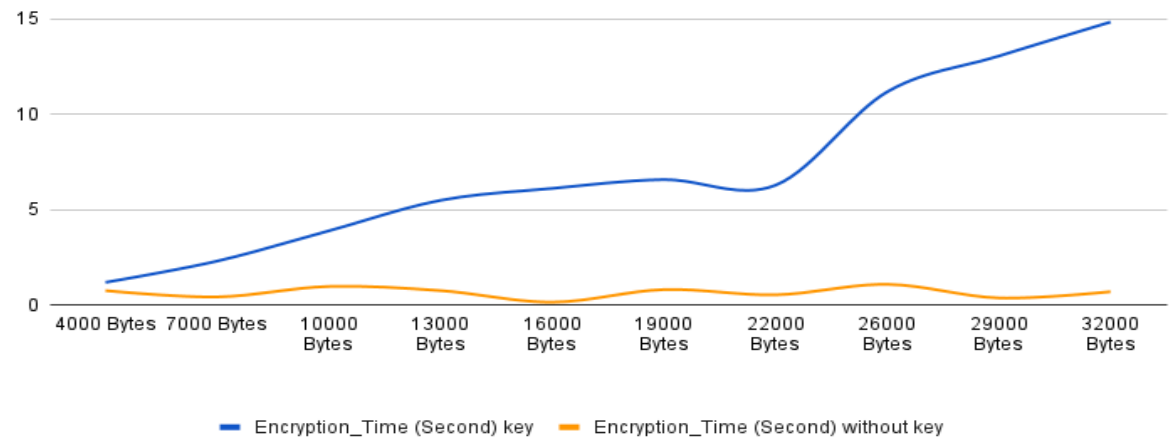
8.5.1. Comparison between Encryption Time of TE and TE using Session Key of .DLL files

8.5.2. Comparison between Decryption Time of TE and TE using Session Key of .DLL files

8.1.1. Comparison between Encryption Time of TE and TE using Session Key of .TXT files

We can from the figure 8.1.1.1 that TE with session key takes more time in encryption than TE technique.

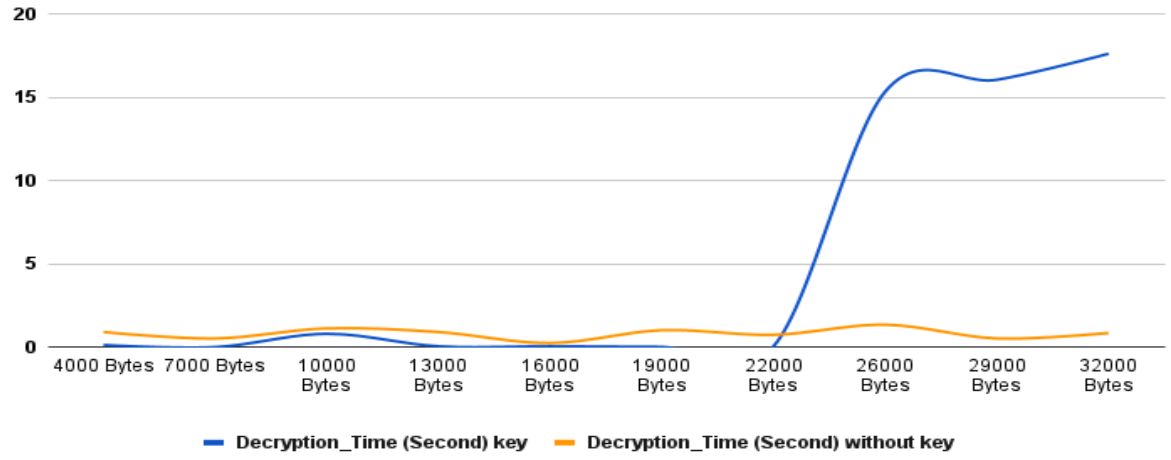
Figure 8.1.1.1



8.1.2. Comparison between Decryption Time of TE and TE using Session Key of .TXT files

We can from the figure 8.1.2.1 that TE with session key more time in decryption than TE technique.

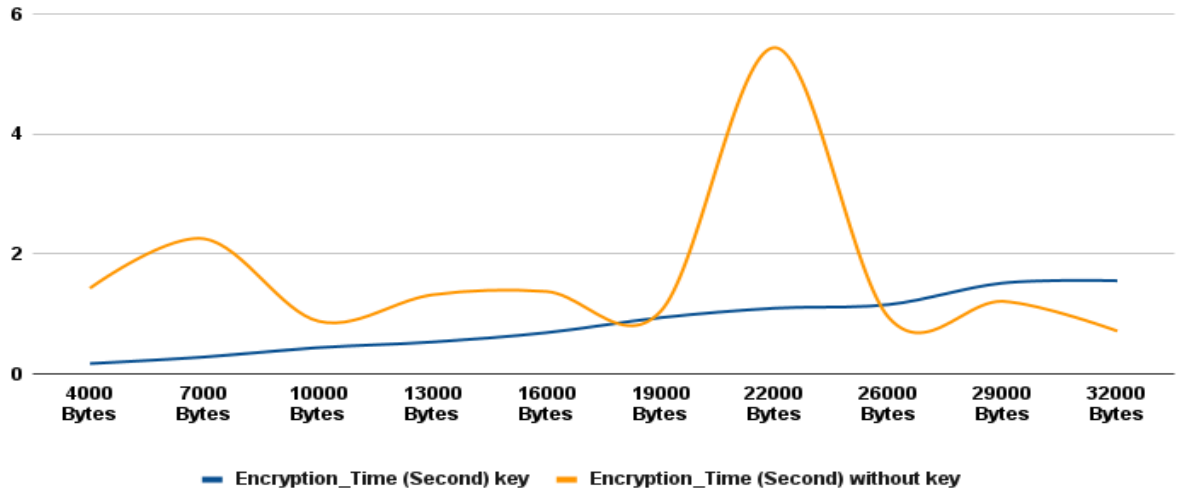
Figure 8.1.2.1



8.2.1. Comparison between Encryption Time of TE and TE using Session Key of .EXE files

We can from the figure 8.2.1.1 that TE with session key more time in encryption than TE technique.

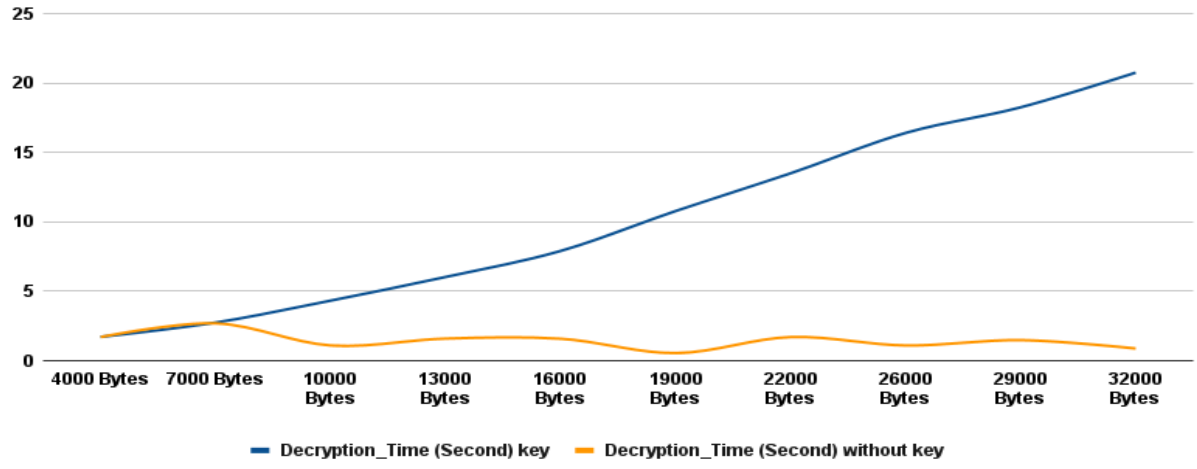
Figure 8.2.1.1



8.2.2. Comparison between Decryption Time of TE and TE using Session Key of .EXE files

We can from the figure 8.2.2.1 that TE with session key has more time in decryption than TE technique.

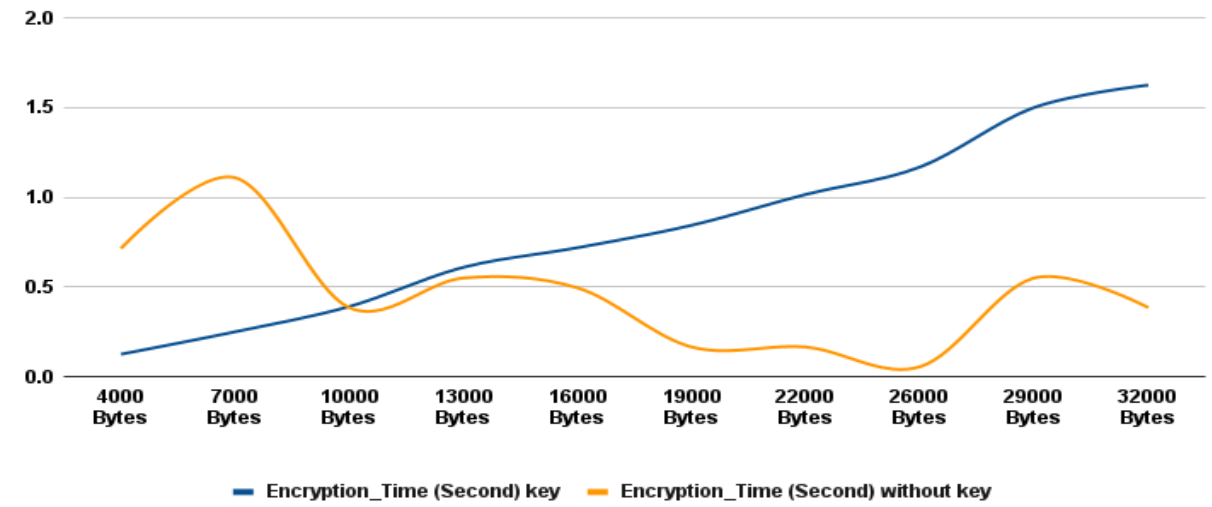
Figure 8.2.2.1



8.3.1. Comparison between Encryption Time of TE and TE using Session Key of .CPP files

We can from the figure 8.3.1.1 that TE with session key takes more time in encryption than TE technique.

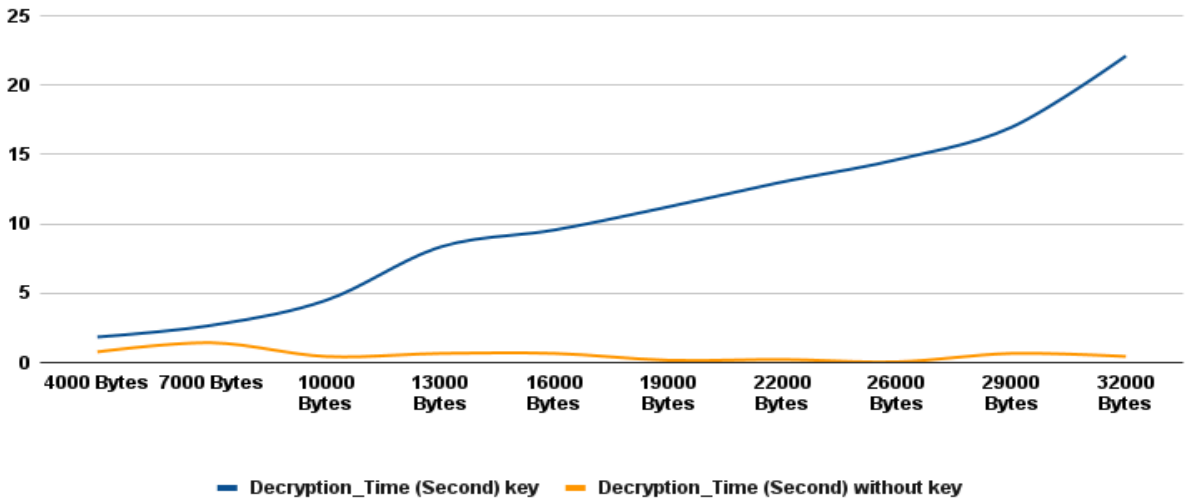
Figure 8.3.1.1



8.3.2. Comparison between Decryption Time of TE and TE using Session Key of .CPP files

We can from the figure 8.3.2.1 that TE with session key has more time in decryption than TE technique.

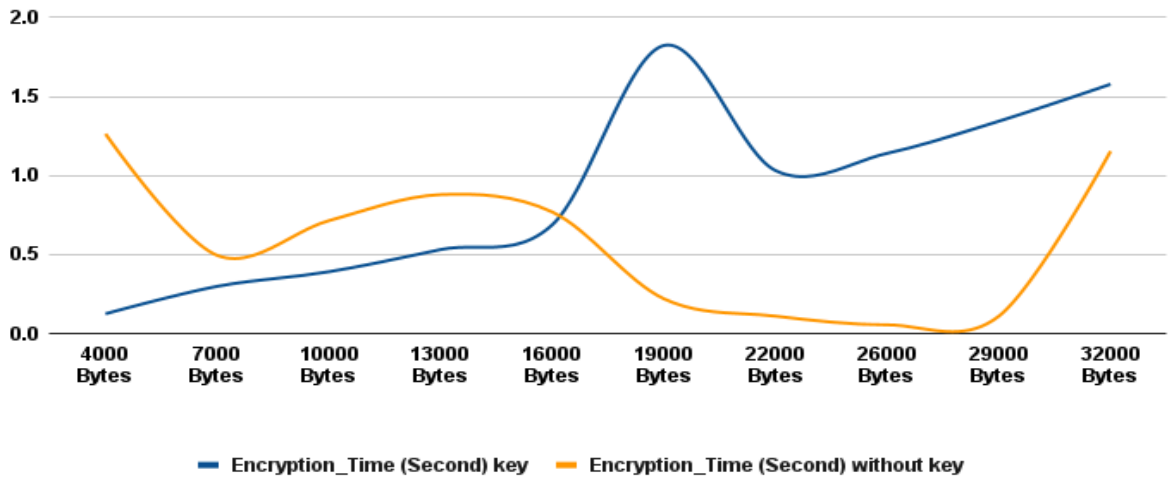
Figure 8.3.2.1



8.4.1. Comparison between Encryption Time of TE and TE using Session Key of .SYS files

We can from the figure 8.4.1.1 that TE with session key takes more time in encryption than TE technique.

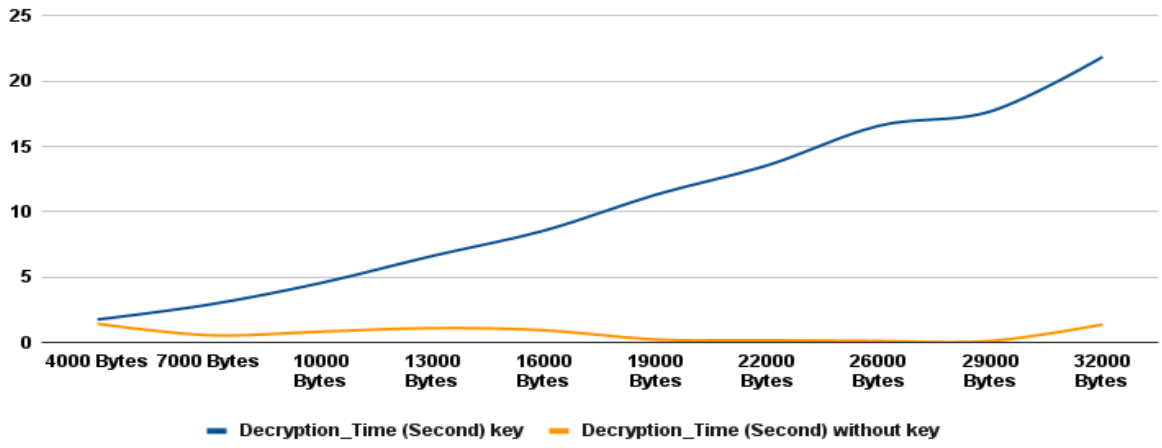
Figure 8.4.1.1



8.4.2. Comparison between Decryption Time of TE and TE using Session Key of .SYS files

We can from the figure 8.4.2.1 that TE with session key has more time in decryption than TE technique.

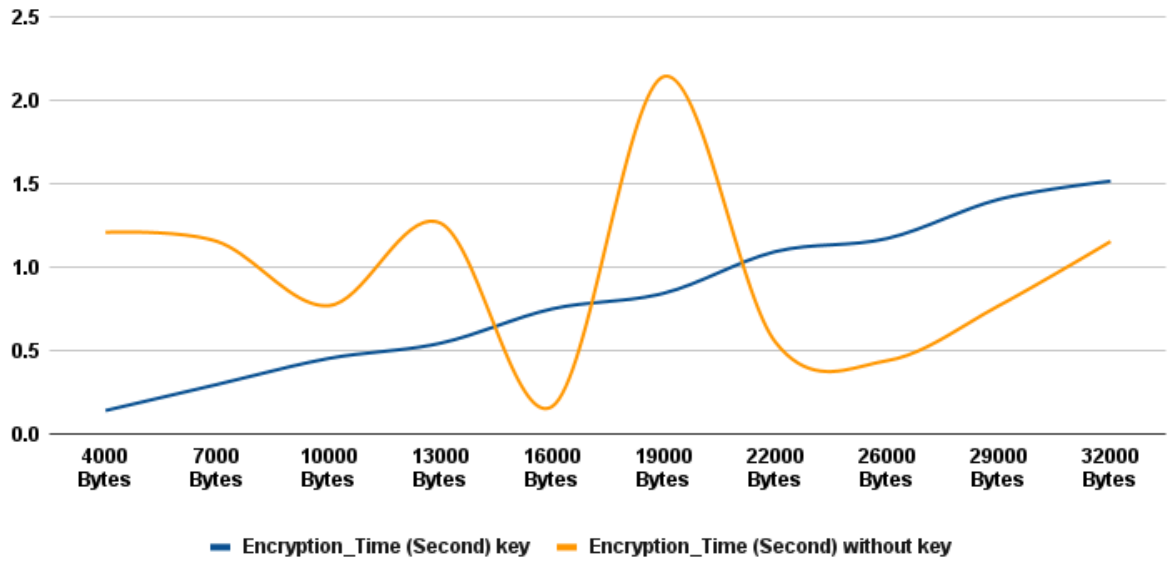
Figure 8.4.2.1



8.5.1. Comparison between Encryption Time of TE and TE using Session Key of .DLL files

We can from the figure 8.5.1.1 that TE with session key takes more time in encryption than TE technique.

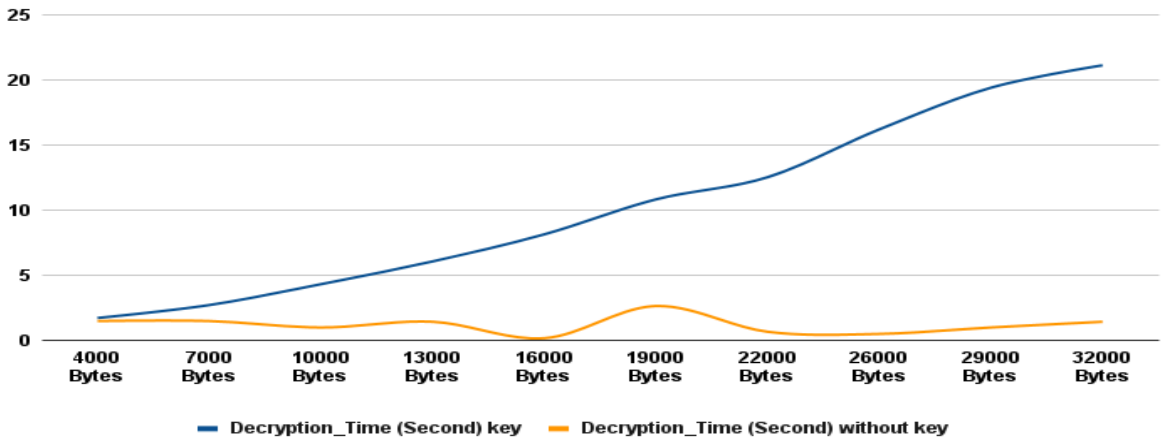
Figure 8.5.1.1



8.5.2. Comparison between Decryption Time of TE and TE using Session Key of .DLL files

We can from the figure 8.5.2.1 that TE with session key has more time in decryption than TE technique.

Figure 8.5.2.1



9. NIST Testing Report :

The National Institute of Standards and Technology (NIST) has developed a suite of statistical tests specifically designed to assess the randomness of cryptographic algorithms. These tests are widely used to verify whether a cryptographic output exhibits properties similar to a truly random sequence, which is crucial for ensuring data security. Implementing NIST tests on the Triangulation Encryption Algorithm ensures that the encryption mechanism produces unpredictable and secure outputs, contributing to a stronger cryptographic system.

```
aditya@DESKTOP-4LJBMRA:/mnt/c/Users/ADITYA/Downloads/sts-2_1_2/sts-2.1.2/sts-2.1.2$ ./assess 100000
        G E N E R A T O R   S E L E C T I O N
        ____________________________________

   [0] Input File                [1] Linear Congruential
   [2] Quadratic Congruential I  [3] Quadratic Congruential II
   [4] Cubic Congruential        [5] XOR
   [6] Modular Exponentiation    [7] Blum-Blum-Shub
   [8] Micali-Schnorr            [9] G Using SHA-1

  Enter Choice: 0

          User Prescribed Input File: data/data22.pi

          S T A T I S T I C A L   T E S T S
          _________________________________

   [01] Frequency                  [02] Block Frequency
   [03] Cumulative Sums            [04] Runs
   [05] Longest Run of Ones        [06] Rank
   [07] Discrete Fourier Transform [08] Nonperiodic Template Matchings
```

[09] Overlapping Template Matchings [10] Universal Statistical
[11] Approximate Entropy [12] Random Excursions
[13] Random Excursions Variant [14] Serial
[15] Linear Complexity

INSTRUCTIONS
Enter 0 if you DO NOT want to apply all of the
statistical tests to each sequence and 1 if you DO.

Enter Choice: 1

P a r a m e t e r A d j u s t m e n t s
---

[1] Block Frequency Test - block length(M): 128
[2] NonOverlapping Template Test - block length(m): 9
[3] Overlapping Template Test - block length(m): 9
[4] Approximate Entropy Test - block length(m): 10
[5] Serial Test - block length(m): 16
[6] Linear Complexity Test - block length(M): 500

Select Test (0 to continue): 0

How many bitstreams? 10

Input File Format:
[0] ASCII - A sequence of ASCII 0's and 1's
[1] Binary - Each byte in data file contains 8 bits of data

Select input mode: 0

Statistical Testing In Progress.....

Statistical Testing Complete!!!!!!!!!!!!

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

---

generator is <data/data22.pi>

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
1	1	3	0	0	2	1	0	1	1	0.534146	10/10	Frequency
1	2	1	0	2	2	1	0	1	0	0.739918	10/10	BlockFrequency
1	1	1	2	1	0	0	2	1	1	0.911413	10/10	CumulativeSums
1	2	0	1	1	1	1	2	1	0	0.911413	10/10	CumulativeSums
0	4	1	1	0	2	0	1	0	1	0.122325	10/10	Runs
0	1	0	4	1	0	1	1	1	1	0.213309	10/10	LongestRun
1	1	0	1	1	1	2	1	0	2	0.911413	10/10	Rank
2	1	0	0	2	1	1	1	2	0	0.739918	10/10	FFT
1	2	1	0	2	2	1	1	0	0	0.739918	10/10	NonOverlappingTemplate
1	3	0	0	3	3	0	0	0	0	0.035174	10/10	NonOverlappingTemplate
3	1	1	0	0	1	0	3	0	1	0.213309	10/10	NonOverlappingTemplate
2	0	1	2	1	0	1	2	1	0	0.739918	10/10	NonOverlappingTemplate
0	2	1	1	0	2	1	1	2	0	0.739918	10/10	NonOverlappingTemplate
1	1	1	1	0	1	0	2	2	1	0.911413	10/10	NonOverlappingTemplate
0	2	1	1	1	2	1	0	1	1	0.911413	10/10	NonOverlappingTemplate
2	1	1	1	1	0	1	3	0	0	0.534146	10/10	NonOverlappingTemplate
1	0	1	1	1	2	1	0	2	1	0.911413	9/10	NonOverlappingTemplate
0	0	2	1	0	1	1	4	1	0	0.122325	10/10	NonOverlappingTemplate
0	2	0	2	0	1	2	1	1	1	0.739918	10/10	NonOverlappingTemplate
2	1	1	1	1	1	0	0	2	1	0.911413	10/10	NonOverlappingTemplate
1	1	3	1	0	1	1	1	1	0	0.739918	10/10	NonOverlappingTemplate
4	0	1	0	0	1	1	1	0	2	0.122325	9/10	NonOverlappingTemplate
1	2	1	0	1	2	1	0	0	2	0.739918	10/10	NonOverlappingTemplate
3	0	1	1	1	1	0	2	1	0	0.534146	10/10	NonOverlappingTemplate
2	1	0	0	2	1	1	2	0	1	0.739918	9/10	NonOverlappingTemplate
0	0	1	3	1	2	0	1	1	1	0.534146	10/10	NonOverlappingTemplate

0	1	1	1	1	0	0	2	3	1	0.534146	10/10	NonOverlappingTemplate
1	1	0	2	1	1	1	0	2	1	0.911413	9/10	NonOverlappingTemplate
0	1	2	0	1	1	1	2	2	0	0.739918	10/10	NonOverlappingTemplate
0	0	0	2	3	1	2	2	0	0	0.213309	10/10	NonOverlappingTemplate
2	0	1	5	0	1	0	0	1	0	0.008879	9/10	NonOverlappingTemplate
1	1	3	1	0	1	1	0	2	0	0.534146	10/10	NonOverlappingTemplate
4	2	1	0	2	0	0	0	1	0	0.066882	10/10	NonOverlappingTemplate
5	1	1	1	0	0	1	0	0	1	0.017912	10/10	NonOverlappingTemplate
2	1	2	1	1	1	0	1	1	0	0.911413	10/10	NonOverlappingTemplate
3	2	0	1	1	1	1	1	0	0	0.534146	9/10	NonOverlappingTemplate
1	1	1	0	1	0	4	1	1	0	0.213309	10/10	NonOverlappingTemplate
0	1	2	1	1	1	1	1	1	1	0.991468	10/10	NonOverlappingTemplate
0	1	1	1	2	0	1	0	2	2	0.739918	10/10	NonOverlappingTemplate
2	1	1	1	1	2	2	0	0	0	0.739918	10/10	NonOverlappingTemplate
0	0	1	0	2	3	0	0	2	2	0.213309	10/10	NonOverlappingTemplate
0	1	1	1	1	1	2	2	0	1	0.911413	10/10	NonOverlappingTemplate
1	1	0	0	0	2	4	2	0	0	0.066882	10/10	NonOverlappingTemplate
0	2	1	1	2	0	2	0	2	0	0.534146	10/10	NonOverlappingTemplate
0	1	0	1	2	2	1	2	0	1	0.739918	10/10	NonOverlappingTemplate
1	0	1	1	0	0	1	2	0	4	0.122325	10/10	NonOverlappingTemplate
2	2	0	0	2	1	2	0	1	0	0.534146	9/10	NonOverlappingTemplate
2	0	1	1	2	1	1	0	1	1	0.911413	10/10	NonOverlappingTemplate
2	0	5	0	0	2	1	0	0	0	0.004301	9/10	NonOverlappingTemplate
0	0	1	1	1	0	2	3	1	1	0.534146	10/10	NonOverlappingTemplate
2	1	2	1	1	0	0	0	1	2	0.739918	9/10	NonOverlappingTemplate
1	0	2	1	0	0	5	1	0	0	0.008879	10/10	NonOverlappingTemplate
0	2	1	1	2	2	0	1	0	1	0.739918	10/10	NonOverlappingTemplate
1	0	1	3	0	2	0	1	0	2	0.350485	10/10	NonOverlappingTemplate
1	2	1	1	1	0	0	1	1	2	0.911413	10/10	NonOverlappingTemplate
2	0	1	0	1	0	0	4	0	2	0.066882	10/10	NonOverlappingTemplate
0	0	0	2	1	1	3	2	1	0	0.350485	10/10	NonOverlappingTemplate
2	1	2	1	0	2	1	0	1	0	0.739918	10/10	NonOverlappingTemplate
3	1	0	2	0	0	1	1	0	2	0.350485	10/10	NonOverlappingTemplate
0	1	2	1	1	0	1	2	0	2	0.739918	10/10	NonOverlappingTemplate

0	1	1	1	3	1	0	2	1	0	0.534146	10/10	NonOverlappingTemplate
1	0	1	0	1	1	1	1	3	1	0.739918	10/10	NonOverlappingTemplate
1	1	3	2	0	0	1	0	2	0	0.350485	10/10	NonOverlappingTemplate
1	0	1	1	0	1	1	1	2	2	0.911413	10/10	NonOverlappingTemplate
2	0	2	0	0	1	2	2	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	1	0	1	1	0	1	3	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	1	1	0	2	0	1	2	0.911413	10/10	NonOverlappingTemplate
1	1	3	1	0	1	1	2	0	0	0.534146	10/10	NonOverlappingTemplate
0	2	1	0	0	0	2	2	1	2	0.534146	10/10	NonOverlappingTemplate
1	1	0	2	2	0	1	0	1	2	0.739918	10/10	NonOverlappingTemplate
0	0	0	0	4	1	3	0	1	1	0.035174	10/10	NonOverlappingTemplate
2	1	3	0	0	1	1	0	0	2	0.350485	10/10	NonOverlappingTemplate
0	1	1	2	0	1	0	0	3	2	0.350485	10/10	NonOverlappingTemplate
0	1	0	1	1	2	0	1	2	2	0.739918	10/10	NonOverlappingTemplate
1	3	0	3	0	2	0	1	0	0	0.122325	10/10	NonOverlappingTemplate
1	2	0	0	2	1	2	1	1	0	0.739918	10/10	NonOverlappingTemplate
0	1	0	2	1	2	1	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	2	0	1	1	1	3	1	1	0	0.534146	10/10	NonOverlappingTemplate
1	2	2	1	1	1	0	0	1	1	0.911413	10/10	NonOverlappingTemplate
1	1	1	4	1	1	0	1	0	0	0.213309	10/10	NonOverlappingTemplate
0	0	0	1	3	1	0	3	2	0	0.122325	10/10	NonOverlappingTemplate
0	1	1	2	1	1	1	2	0	1	0.911413	10/10	NonOverlappingTemplate
1	2	1	0	2	2	1	1	0	0	0.739918	10/10	NonOverlappingTemplate
3	1	2	1	0	0	0	1	0	2	0.350485	10/10	NonOverlappingTemplate
1	0	1	2	2	1	0	1	1	1	0.911413	10/10	NonOverlappingTemplate
0	1	1	1	2	1	3	1	0	0	0.534146	10/10	NonOverlappingTemplate
0	3	1	0	2	2	1	0	1	0	0.350485	10/10	NonOverlappingTemplate
1	0	0	0	3	3	1	1	0	1	0.213309	10/10	NonOverlappingTemplate
2	0	1	1	1	1	1	1	1	1	0.991468	10/10	NonOverlappingTemplate
1	1	1	1	0	1	0	2	1	2	0.911413	10/10	NonOverlappingTemplate
1	2	0	0	0	3	2	1	0	1	0.350485	10/10	NonOverlappingTemplate
0	1	0	1	0	0	1	2	3	2	0.350485	10/10	NonOverlappingTemplate
1	0	2	1	3	0	1	2	0	0	0.350485	10/10	NonOverlappingTemplate
0	2	2	1	1	1	1	0	1	1	0.911413	10/10	NonOverlappingTemplate

0	1	0	3	0	0	1	3	1	1	0.213309	10/10	NonOverlappingTemplate
2	2	3	1	0	0	1	1	0	0	0.350485	10/10	NonOverlappingTemplate
2	1	0	0	0	0	0	1	2	4	0.066882	10/10	NonOverlappingTemplate
1	1	0	0	1	3	1	1	0	2	0.534146	10/10	NonOverlappingTemplate
0	1	0	2	1	0	4	1	0	1	0.122325	10/10	NonOverlappingTemplate
0	4	0	0	1	1	1	0	2	1	0.122325	10/10	NonOverlappingTemplate
1	1	0	1	1	1	1	2	0	2	0.911413	10/10	NonOverlappingTemplate
2	2	2	0	2	0	1	0	1	0	0.534146	10/10	NonOverlappingTemplate
0	1	3	1	0	1	2	0	0	2	0.350485	10/10	NonOverlappingTemplate
1	2	0	0	1	1	1	0	1	3	0.534146	10/10	NonOverlappingTemplate
2	0	1	1	1	1	2	0	0	2	0.739918	10/10	NonOverlappingTemplate
0	2	1	1	0	1	2	2	0	1	0.739918	10/10	NonOverlappingTemplate
0	0	4	1	1	0	1	1	0	2	0.122325	10/10	NonOverlappingTemplate
1	1	1	0	2	2	0	1	2	0	0.739918	10/10	NonOverlappingTemplate
2	1	1	1	0	1	1	0	0	3	0.534146	9/10	NonOverlappingTemplate
0	1	0	1	0	1	1	0	3	3	0.213309	10/10	NonOverlappingTemplate
1	1	2	1	1	1	0	2	0	1	0.911413	10/10	NonOverlappingTemplate
0	2	0	0	1	0	3	1	2	1	0.350485	10/10	NonOverlappingTemplate
3	0	1	1	1	0	1	1	1	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	1	1	0	0	2	1	2	0.911413	10/10	NonOverlappingTemplate
2	2	0	0	0	1	2	1	1	1	0.739918	10/10	NonOverlappingTemplate
0	0	1	2	2	0	2	1	2	0	0.534146	10/10	NonOverlappingTemplate
1	1	0	2	1	1	0	2	1	1	0.911413	10/10	NonOverlappingTemplate
0	2	0	1	3	1	1	2	0	0	0.350485	10/10	NonOverlappingTemplate
2	1	1	1	2	0	0	2	1	0	0.739918	9/10	NonOverlappingTemplate
0	1	2	0	1	3	1	0	2	0	0.350485	10/10	NonOverlappingTemplate
1	1	0	1	2	0	3	1	1	0	0.534146	10/10	NonOverlappingTemplate
2	0	1	1	0	0	1	0	2	3	0.350485	10/10	NonOverlappingTemplate
1	0	0	1	0	0	1	4	2	1	0.122325	10/10	NonOverlappingTemplate
0	2	0	3	1	3	0	0	0	1	0.122325	10/10	NonOverlappingTemplate
0	0	2	1	2	2	1	1	0	1	0.739918	10/10	NonOverlappingTemplate
1	1	1	3	0	0	2	1	0	1	0.534146	10/10	NonOverlappingTemplate
1	1	1	0	1	1	2	0	3	0	0.534146	10/10	NonOverlappingTemplate
3	0	1	3	1	1	0	0	0	1	0.213309	10/10	NonOverlappingTemplate

1	1	3	1	0	2	0	1	0	1	0.534146	10/10	NonOverlappingTemplate
1	0	0	2	3	1	1	1	0	1	0.534146	10/10	NonOverlappingTemplate
0	0	2	0	2	1	1	1	2	1	0.739918	10/10	NonOverlappingTemplate
1	3	1	0	1	1	0	0	3	0	0.213309	10/10	NonOverlappingTemplate
1	0	2	2	1	1	1	1	0	1	0.911413	10/10	NonOverlappingTemplate
2	2	1	1	0	2	1	0	1	0	0.739918	10/10	NonOverlappingTemplate
1	1	4	1	1	0	0	1	0	1	0.213309	10/10	NonOverlappingTemplate
0	3	0	0	0	0	1	1	2	3	0.122325	10/10	NonOverlappingTemplate
0	0	0	1	4	1	1	2	0	1	0.122325	10/10	NonOverlappingTemplate
4	2	1	1	0	1	0	0	1	0	0.122325	10/10	NonOverlappingTemplate
0	1	2	2	0	1	0	1	1	2	0.739918	10/10	NonOverlappingTemplate
0	1	2	1	2	1	1	1	1	0	0.911413	10/10	NonOverlappingTemplate
1	0	1	0	0	0	2	0	1	5	0.008879	10/10	NonOverlappingTemplate
2	0	3	2	0	0	1	0	2	0	0.213309	9/10	NonOverlappingTemplate
4	1	2	1	1	0	0	0	0	1	0.122325	9/10	NonOverlappingTemplate
2	2	1	1	0	2	0	1	0	1	0.739918	10/10	NonOverlappingTemplate
0	1	0	1	5	1	0	1	0	1	0.017912	10/10	NonOverlappingTemplate
0	0	0	2	0	1	3	1	2	1	0.350485	10/10	NonOverlappingTemplate
0	1	1	1	0	1	3	1	1	1	0.739918	10/10	NonOverlappingTemplate
1	0	1	1	1	2	2	0	2	0	0.739918	10/10	NonOverlappingTemplate
2	0	0	0	0	1	2	2	1	2	0.534146	10/10	NonOverlappingTemplate
1	0	2	1	0	1	2	0	1	2	0.739918	10/10	NonOverlappingTemplate
1	0	1	0	1	3	2	0	1	1	0.534146	10/10	NonOverlappingTemplate
1	0	0	0	2	2	1	3	1	0	0.350485	10/10	NonOverlappingTemplate
2	2	1	0	0	2	1	0	1	1	0.739918	10/10	NonOverlappingTemplate
3	2	0	3	0	1	0	0	1	0	0.122325	10/10	NonOverlappingTemplate
2	2	0	2	0	2	2	0	0	0	0.350485	10/10	NonOverlappingTemplate
0	1	1	2	1	1	1	2	0	1	0.911413	10/10	NonOverlappingTemplate
4	0	1	2	0	1	0	0	0	2	0.066882	9/10	OverlappingTemplate
10	0	0	0	0	0	0	0	0	0	0.000000 *	0/10 *	Universal
0	1	1	2	0	1	3	1	1	0	0.534146	10/10	ApproximateEntropy
0	2	0	0	0	0	0	0	0	0	----	2/2	RandomExcursions
0	0	0	0	1	0	1	0	0	0	----	2/2	RandomExcursions
0	0	0	0	1	0	0	1	0	0	----	2/2	RandomExcursions

0	0	0	0	0	0	1	1	0	0	----	2/2	RandomExcursions
0	0	1	0	0	0	0	1	0	0	----	2/2	RandomExcursions
0	0	1	0	0	0	0	0	0	1	----	2/2	RandomExcursions
0	0	0	1	0	0	1	0	0	0	----	2/2	RandomExcursions
0	1	0	0	1	0	0	0	0	0	----	2/2	RandomExcursions
0	0	0	1	0	0	0	1	0	0	----	2/2	RandomExcursionsVariant
0	0	0	1	0	0	0	1	0	0	----	2/2	RandomExcursionsVariant
0	0	0	1	0	0	0	0	0	1	----	2/2	RandomExcursionsVariant
0	0	0	1	0	1	0	0	0	0	----	2/2	RandomExcursionsVariant
0	0	0	0	1	1	0	0	0	0	----	2/2	RandomExcursionsVariant
0	0	0	0	2	0	0	0	0	0	----	2/2	RandomExcursionsVariant
0	0	0	1	1	0	0	0	0	0	----	2/2	RandomExcursionsVariant
0	0	0	0	0	1	1	0	0	0	----	2/2	RandomExcursionsVariant
0	0	0	0	0	2	0	0	0	0	----	2/2	RandomExcursionsVariant
0	1	0	0	0	0	0	0	0	1	----	2/2	RandomExcursionsVariant
0	1	0	0	0	0	0	0	1	0	----	2/2	RandomExcursionsVariant
0	1	0	1	0	0	0	0	0	0	----	2/2	RandomExcursionsVariant
0	1	1	0	0	0	0	0	0	0	----	2/2	RandomExcursionsVariant
0	1	0	1	0	0	0	0	0	0	----	2/2	RandomExcursionsVariant
0	0	2	0	0	0	0	0	0	0	----	2/2	RandomExcursionsVariant
0	0	1	1	0	0	0	0	0	0	----	2/2	RandomExcursionsVariant
0	0	1	0	1	0	0	0	0	0	----	2/2	RandomExcursionsVariant
0	1	0	0	0	0	1	0	0	0	----	2/2	RandomExcursionsVariant
3	0	2	1	0	0	1	0	1	2	0.350485	9/10	Serial
2	2	1	1	0	2	0	0	0	2	0.534146	9/10	Serial
2	2	1	0	0	1	1	0	2	1	0.739918	10/10	LinearComplexity

- -

The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately = 8 for a sample size = 10 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 1 for a sample size = 2 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- -

10. Conclusion :

In conclusion, the Triangular Encryption (TE) technique offers a significant advancement over traditional methods like RSA, with its unique geometric transformations providing enhanced security and resistance to common cryptographic attacks. TE ensures data integrity and offers a versatile, scalable solution for various applications, making it a promising candidate for addressing evolving cyber threats. The future scope of TE is vast, including securing real-time voice communications, protecting images through encryption and digital watermarking, and ensuring the confidentiality of video content in streaming services and video conferencing. As research and development continue, TE is poised to become a standard in cryptographic practices, offering robust solutions for securely encrypting voice, images, and videos, thereby enhancing overall data security and privacy.

11. Future Works :

TE can be used to encrypt text messages, ensuring that communications remain private and protected from unauthorized access. The TE (Triangulation Encryption) algorithm can be further tested on multimedia formats like images, videos and audios. After testing it we can use the algorithm on multimedia-sharing platforms, TE can ensure that shared content remains confidential and is accessible only to authorized users. This is particularly important for platforms dealing with sensitive or proprietary media content.

12. Reference :

1. Bruce Schneier, Applied Cryptography. Second Edition, Protocols, Algorithms, and Source Code in C, John Wiley & Sons Inc., 1996.

2. L.M. Adleman, C. Pomerance and R. S. Rumeley, "On Distinguishing Prime Numbers from Composite Numbers", Annals of Mathematics, v. 117, n. 1, 1983, pp. 173-206.

3. H. Fiestel, "Cryptography and Computer Privacy," Scientific American, v. 228, n.5, May 1973, pp. 15-23.

4. G. B. Agnew, "Random Sources of Cryptographic Systems," Advances in Cryptology: EURCRYPT '87 Proceedings, Springer Verlag, 1988, pp. 77- 81.

5. S. G. AKI and H. Meijer, "A Fast Pseudo-Random Permutation Generator with Applications to Cryptology," Advances in Cryptology: - EURCRYPT 84 Proceedings, Springer Verlag, 1985, pp. 269-275.

6. W. Alexi, B.Z. Chor, O. Goldreich and C. P. Schnorr. "RSA and Rabin Functions: Certain Parts are as Hard as the Whole", SIAM Journal on Computing, v. 17, n.2, Apr 1988, pp. 194-209.

7. W. Alexi, B.Z. Chor, O. Goldreich and C. P. Schnorr. "RSA and Rabin Functions: Certain Parts are as Hard as the Whole" Proceedings of the 25th IEEE Symposium on the Foundation of Computer Science, 1984, pp. 449-457.

8. R. J. Anderson, "Why Cryptosystems Fail," 1" ACM Conference on Computer and Communications Security ACM Press, 1993, pp. 215-227.

9. R. J. Anderson, "Why Cryptosystems Fail," Communications of the ACM, v.37, n. 11, Nov 1994, pp. 32-40.

10. J. Anderson, and R. Needham, "Robustness of Principles for Public Key Protocols," Advances in Cryptology: - EURCRYPT '95 Proceedings, Springer Verlag, 1995.

11. C. Asmuth and J. Bloom, "A Modular Approach to Key Safeguarding," IEEE Transactions on Information Theory, v.IT 29, n. 2. Mar 1983, pp. 208-210..

12. S. K. Banerjee, "High Speed Implementation of DES," Computers & Security, v.1, 1982, pp. 261-267.

13. E. Biham and A. Biryukov, "How to Strengthen DES Using Existing Hardware," Advances in Cryptology ASIACRYPT '94 Proceedings, Springer-Verlag, 1995.

14. E. Biham and A. Shamir, "Differential Crytosystems," Advances in Cryptology Springer-Verlag, 1991, pp. 2-21. Cryptanalysis of DES-like CRYPTO '90 Proceedings,

15. E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Crytosystems," Journal of Cryptology, v. 4, n. 1, 1991, pp. 3-72.
16. E. Biham and A. Shamir, "Differential Cryptanalysis of the Full 16-Round DES," Advances in Cryptology CRYPTO 92 Proceedings, Springer- Verlag, 1993, pp. 487- 496.
17. G. Brassard, "A Note on Complexity of Cryptography." IEEE Transactions on Information Theory, v. IT 25, n. 2, Mar 1979, pp. 232-233.
18. G. Brassard, "Relativized Cryptography," Proceedings of the IEEE 20th Annual Symposium on the Foundations of Computer Science, 1979, pp.383- 391.
19. G. Brassard, ""Relativized Cryptography," IEEE Transactions on Infor Theory, v. IT 29, n. 6, Nov 1983, pp. 877-894.
20. E. F. Brickell, and A.M. Odlyzko, "Cryptanalysis: A Survey of Results," Proceedings of the IEEE v. 76, n. 5, May 1988. pp. 578-593.
21. D. Coppersmith, "Data Encryption Standard (DES) and its Strength against Attacks," IBM Journal of Research and Development, v. 38, n. 3, May 1994, pp. 243-250.
22. C. Cauvreur and J.J. Quisquater, "An Introduction to Fast Generation of Large Prime Numbers," Philips Journal Research, v. 37, n. 5-6, 1982, pp. 231-264.
23. C. Cauvreur and J.J. Quisquater, "An Introduction to Fast Generation of Large Prime Numbers," Philips Journal Research, v. 38, 1983, pp. 77.
24. J. Daemen, and J. Vandewalle, "Block Cipher Based on Modular Arithmetic," Proceedings of Third Symposium on State and Progress of Research in Cryptography, Rome Italy, 15-16 Feb 1993, pp. 80-89.
25. J. Daemen, R. Govaerts, and J. Vandewalle, "A New Approach to Block Cipher Design," Fast Software Encryption, Cambridge Security Workshop Proceedings, Springer-Verlag, 1994, pp. 18-32.
26. D. E. Denning, "Data Encryption Standard: Fifteen Years of Public Scrutiny", Proceedings Sixth Annual Computer Security Applications Conference, IEEE Computer Society Press, 1990.
27. A. Di Porto and W. Wolfowicz, "VINO: A Block Cipher Including Variable Permutations," Fast Software Encryption Cambridge Security Workshop Proceedings, Springer-Verlag, 1994, pp. 205-210.
28. S. Even and O. Goldreich, "DES-Like Functions Can Generate the Alternating Group," IEEE Transactions on Information Theory, v. IT-29. n. 6, Nov 1983, pp. 863-865.
29. S. Even and O. Goldreich, "On the Power of Cascade Ciphers," ACM Transactions on Computer Systems, v. 3, n. 2, May 1985, pp. 108-116.
30. H. Gustafson, E. Dawson and B. Caelli, "Comparison of Block Ciphers," Advances in Cryptology AUSCRYPT '90 Proceedings. Springer-Verlag 1990, pp. 208-220.
31. X. Lai and J. Massey, "A Proposal for New Block Encryption Standard," Advances in Cryptology EUROCRYPT '90 Proceedings, Springer-Verlag, 1991, pp. 389-404.

32. Y. Ohnishi "A Study of Data Security," Master's Thesis Tohuku University, Japan 1988.
33. R.L. Rivest, "A Description of a Single-Chip Implementation of RSA Cipher," LAMDA Magazine, v.1, n. 3, 1980, pp. 14-18.
34. M. J. B. Robshaw, "Block Ciphers" Technical Report TR-601, RSA Laboratories, Jul 1994.
35. B. Schneier, "One-Way Hash Functions," Dr. Dobb's Journal, v.16, n. 9, Sep 1991, pp. 148-151.
36. B. Schneier, "Description of a New Variable-length Key, 64-bit Block Cipher (Blowfish)," Fast software Encryption, Cambridge Security Workshop Proceedings, Springer-Verlag, 1994, pp. 191-204.
37. N. G. Das, "Statistical Methods in Commerce, Accountancy & Economics(Part II)", M. Das & Co.
38. Mandal J. K. and Dutta S., "A Space-Efficient Universal Encoder for S Transmission", International Conference on Modelling and Simulation 2000- Egypt), Cairo, April 11-14, 2000, pp-193-201.
39. Mandal J. K., Mal S., Dutta S., A 256 Bit Recursive Pair Parity Encoder for Encryption, accepted for publication in AMSE Journal, France, 2003