International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-7 , April, 2014

An Approach of Bit-level Private-key Encryption Scheme based on Alphabetic Group, User-defined Operator and Palindrome Number in Selective Mode

Ramkrishna Das[1],Sukhendu Patra[2], Saurabh Dutta[3]

Abstract – Private-key cryptography is a cryptographic system that uses the same secret key to encrypt and decrypt messages. The problem with this method is transmitting the secret key to the receiver who needs it without being intercepted. Many of the existing private-key cryptography systems are complex and not up to the mark with respect to security, as the distribution of the private-key without interpretation are very hard to achieve. In this paper, we have focused on the secret procedure to retrieve secret value from the private-key rather than securing the actual private-key value. The encryption is done by the secret value derived from the private-key. The secret value is being derived by combining n^{th} (consonant, sequence, special character, vowel or semivowel), user-defined operator (+,-,*, /) and n^{th} palindrome (derived from a user defined base value towards its forward or backward direction). As per the user defined sequence. Thus an attempt is made to enhance the security.

Index Terms – N^{th} (consonant, vowel or semi vowel, special character, and palindrome number), Private-key encryption, Stream cipher**.**

I. INTRODUCTION

Cryptography is the practice and study of techniques for secure transmission of information between receiver and sender in presence of other parties.

Private-key cryptography refers to an encryption method where both the sender and the receiver use either the identical secret key or two keys derivable from each other. Fig-1 represents a private-key encryption method that uses the same secret key for encryption and decryption.

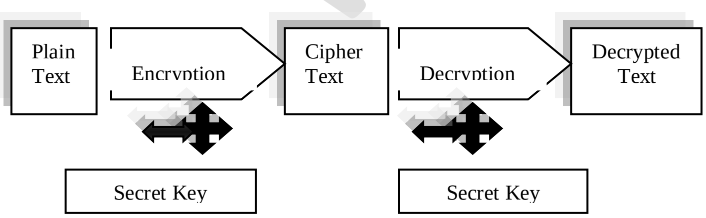

Figure 1: A Private-key Encryption Scheme

The traditional demerit of private-key encryption technique is to distribute the private-key securely. Noticeably, many of the existing private-key encryption systems suffer from lack of security.

Here we have developed a procedure, which is responsible to retrieve the secret value from the private-key. The value is being used both for encryption and decryption. Searching N^{th} (consonant, sequence, special character, vowel or semi vowel), operator (+,-,*, /) from key value. Again searching N^{th} palindrome from a user defined base value towards its forward or backward direction. As per the user defined sequence. Herein lays the attempt to increase security, as we focus on securing the retrieving procedure rather than directly the private-key value. Secret value can't be retrieved without the knowledge of the retrieving procedure [2] [3] [4]

In this paper, Section-II describes the encryption process; Section-III describes the decryption process. Experimental results are being described in Section-IV and Section-V draws the conclusion.

Ramkrishna Das[1], Assistant Professor, Department of Computer Applications, Haldia Institute of Technology, Haldia, WB, INDIA, 9933718033, ramkrishnadas9@gmail.com

Suukhendu Patra[2], PG Student ,Depatrment of computer Science, Vidyasagar University, Paschim Medinipur,WB,INDIA,9563009207, patrasukhendu03@gmail.com

Dr. Saurabh Dutta[3], Professor and Head, Department of Computer Applications, Dr. B. C. Roy Engineering College, Durgapur-713206, WB, INDIA, 9433411450, saurabh.dutta@bcrec.org

II. ENCRYPTION PROCESS

A palindrome number is a symmetrical number meaning that the number remains the same even when the digits are reversed. An example of a palindrome number *is 11.*

If k is odd, formula = $2(n - l)[(n - 1)(k+1)/2 - 1] - (n - 1)(k+1)/2\ n - 2.$

We have used the English alphabetic groups like vowel, semi-vowel, consonant and special character. A user defined sequence is being used to combine the results for generating the secret value from the private key. Step A, step B, step C, step D sequentially describes the encryption process [1] [2] [3].

Plain Text Formation

www.ijitam.org

Let 'b' is a character which is present in the inputted file. Fig-2 represents its 8-bit binary representation through an array PLAINTEXT with dimension 8.

Plaintext1--Plaintext8

0	1	1	0	0	0	1	0
1st bit	2nd bit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

Figure 2: Formation of Plain Text

Step-A.1 Read one character at a time from the inputted file till we reached to the end of the file. Convert each character into 8-bit binary representation and store the value into the array PLAINTEXT with dimension 8.

A. key Generation

The size of the private-key is 40 bits having 8 blocks. 1st 5-bit block represents the choice nth consonant and 2nd 5-bit block represents the choice nth special character. 3rd 3-bit block represents the choice nth vowel or semivowel. 4th 5-bit block the choice nth sequence. 5th 6-bit block represents the operator (+,-,*, /).6th 2-bit block represents the forward or backward movement from base value (1/0).7th 4-bit block represents the nth term for Palindrome number. 8th 10-bit block represents the user defined base value.Fig-3 represents block diagram of the 40-bit private-key.

1st block	2nd block	3rd block	4th block
Nth term for consonant	Nth term for special character	Nth term for Vowel or semivowel	Nth term for sequence
5-bit	5-bit	3-bit	5-bit

5th block	6th block	7th block	8th block
User defined operator	Forward or Backward movement from base value	Nth term for Palindrome number	User defined base value
6-bit	2-bit	4-bit	10-bit

Figure-3: Block Diagram of 40-bit Private-key

Step B.1 Read the user inputs for 8 blocks from the user. Convert those input values into corresponding bit size of their respective blocks and store the values in an array KEY with dimension 40.

C. Formation of Secret Value from Private-key for Encryption

Step C.1 The value of the 1st block, 2nd block, 3rd block is holds the nth term of consonant, special character, vowel and semi-vowel respectively. A user-defined sequence value is stored in the 4th block of the private-key. The user-defined operator (+,-,*, /) value is stored in the 5th block of the private-key. The value of the 7th block is being used to determine the nth term for Palindrome number from base value. The value of the 5nd block determines the forward or backward movement from the base value. The value of the 8th block holds base value. Searching of Nth palindrome would be done by making a forward (1) or backward (0) movement from the base value. Where N is a positive integer in the range of (1<=N<=16).The user defined sequence value stored in 4rd block is being used to combine the results (Nth (consonant, special character, vowel and semi-vowel, intermediate value)) together.

Step C.2 Determine the Nth Palindrome Number by making a forward or backward movement from the user-defined base value and generate the corresponding nth consonant, special character, vowel or semi-vowel and generate the intermediate value.

Step C.3 Sequence wise combined the 40 bit Value generation.

Example:-The example demonstrates the private-key formation and the secret value generation procedure. Fig-4 represents the bit wise representation of a private-key for some specific value.

----------Nth consonant (11) ------- ----Nth special character (18) --------

1	2	3	4	5	6	7	8	9	10
0	1	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**0**

---------------5-bit-------------------- ----------------------5-bit------------

--Nth V & S.V(5)---- -------- Nth sequence(12)------------- ------------

11	12	13	14	15	16	17	18	19	20
1	**0**	**1**	**0**	**1**	**1**	**0**	**0**	**1**	**0**

---3-bit------------- ------------5-bit --------------------- ---------

Operator (+) (43) ------------- ----F/B(1)----- ----Nth Palindrome(8)------

21	22	23	24	25	26	27	28	29	30
1	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**

6-bit--------------------------- --2-bit--------- ------------4-bit----------------

-----------------------------------Base value (10) -------------------------------------

31	32	33	34	35	36	37	38	39	40
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**

-----------------------------------10-bit---

Figure 4: 40- bit Representation of Private-key for a Specific Value.

User-defined base value is 10 which is store in 8th block. 6th block of the private-key is 1 so the palindrome number is taken by making a forward movement. 1st block of the private-key is 11, so the corresponding consonant is "N", 2nd block of the private-key is 18, so the corresponding special character is "=". 3rd block of the private-key is 5, so the corresponding vowel or semi-vowel is "U". As the 6th block value determines the forward movement so we have to make a forward movement (11,12,13,14...) from base value which is 10 .7th block represents the Nth term which is 8 . So we take the 8th palindrome number with a forward movement from base value. The number is 88. 5th block of the private-key is 43, that is the numeric decimal value of "+". We combine all this result as per the combination no which is 12.The combination number is store in 4rd block. The definition of the combination is (Nth special character +intermediate value + Nth vowel or

www.ijitam.org

semi-vowel + Nth consonant). The result is combining as per the combination value. The value is "=98UN".
Fig-5 represents the 48-bit representation of secret value.

1	2	3	4	5	6	7	8	9	10	11	12
0	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**

-------------------------8- bit---------------------------- -------------------------
DV1---DV8 -----------------------------

13	14	15	16	17	18	19	20	21	22	23	24
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

------------8- bit------------- ----------------------------8- bit------------------------
------------------------DV16 ---DV24

25	26	27	28	29	30	31	32	33	34	35	36
0	1	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	0	**1**

----------------------8-bit--------------------------- -------------8-bit-------
DV25--DV32--------------------------

37	38	39	40	41	42	43	44	45	46	47	48
0	**1**	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**1**	**0**

------------------------- ------------------------8-bit------------------------
---------------------DV40---DV48

Figure 5: 48- bit Representation of Secret Value derived from Private-key.

D. XOR operation and Formation of Cipher Text

Plain text is being encrypted by the 6 blocks of the secret value cumulatively where the block size is 8 bits. Bitwise XOR operation is being performed between the plain text and the secret value Fig-6 represents the encryption process

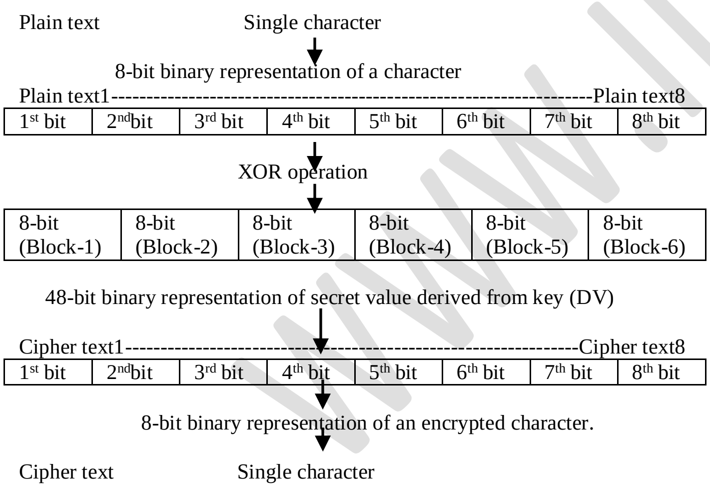

Figure 6: XOR operation between Plain Text and Secret Value.

Cumulative XOR operation is performed between plain text and the secret value. 6th block, 5th block,4th block 3rd block, 2nd block and the 1st block of the secret value (DV) are used for XOR respectively After 6 times, we get the binary value of corresponding ASCII code of a final encrypted character. In this way cipher text file is generated and sent to the receiver with the secret private-key file. Fig-7 demonstrate the total XOR procedure between plain text and the secret value where IET means intermediate encrypted text

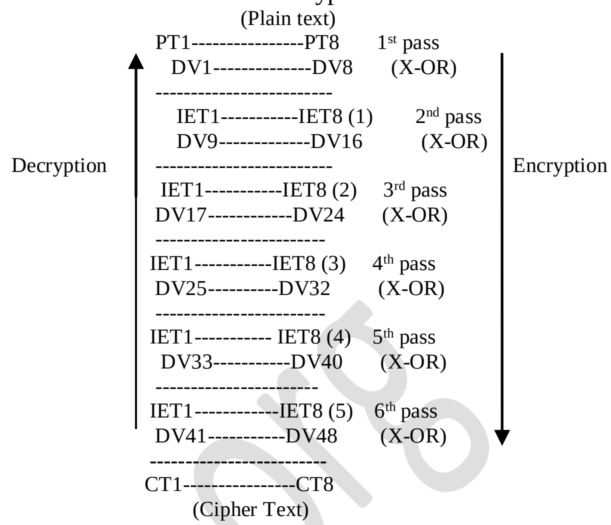

Figure 7: Block wise Cumulative XOR operation between Plain Text and Secret Value derived from the Private-key.

Step D.1 Perform XOR operation between the array PLAINTEXT and DERIVEDVALUE. We consider 1st block, 2nd block , 3rd block, 4th block, 5th block and last block of the array DERIVEDVALUE for XOR operation in 1st pass, 2nd pass, 3rd pass, 4th pass,5th pass,6th pass respectively. Intermediate result is stored into an array IET with dimension 8. And final result is stored into array ENCRYPTED with dimension 8.Corresponding ASCII code is generated from of the array ENCRYPTED and from there we get the encrypted character 'b' from plain text whose ASCII value is 98. We generate the plain text in step-A and secret value in step-C. Now we perform the XOR operation between the plain text and the secret value. Fig-8 represents the XOR procedure

Plaintext1---Plaintext8

0	**1**	**1**	**0**	**0**	**0**	**1**	**0**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV1---DV8

0	**0**	**1**	**1**	**1**	**1**	**0**	**1**
1th bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

Intermediate Encrypted Text (IET (1))

0	**1**	**0**	**1**	**1**	**1**	**1**	**1**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV9--DV16

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
9th bit	10ndbit	11rd bit	12th bit	13th bit	14th bit	15th bit	16th bit

Intermediate Encrypted Text (IET (2))

0	**1**	**0**	**1**	**1**	**1**	**1**	**1**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

www.ijitam.org

DV17---DV24

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
17th bit	18ndbit	19rd bit	20th bit	21th bit	22th bit	23th bit	24th bit

↓

Intermediate Encrypted Text (IET (3))

0	**1**	**0**	1	**1**	**1**	1	**1**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV25---DV32

0	**1**	**1**	**0**	**0**	**0**	**1**	**0**
25th bit	26ndbit	27rd bit	28th bit	29th bit	30th bit	31th bit	32th bit

↓

Intermediate Encrypted Text (IET (4))

0	**0**	**1**	**1**	**1**	**1**	**0**	**1**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV33---DV40

0	**1**	**0**	**1**	**0**	**1**	**0**	**1**
33th bit	34ndbit	35th bit	36th bit	37th bit	38th bit	39th bit	40th bit

↓

Intermediate Encrypted Text (IET (5))

0	**1**	**1**	**0**	**1**	**0**	**0**	**0**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV41---DV48

0	**1**	**0**	**0**	**1**	**1**	**1**	**0**
41th bit	42ndbit	43rd bit	44th bit	45th bit	46th bit	47th bit	48th bit

↓

Cipher Text

0	**0**	**1**	**0**	**0**	**1**	**1**	**0**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

Figure 8: Generation of Cipher Text by Block Wise Cumulative XOR operation between Plain Text and Secret Value.

ASCII value corresponding to the cipher text is 38 and the character corresponding to that ASCII is '&' In this way all the characters are encrypted and stored into the cipher text file. The encrypted file is sent to the receiver with the secret private-key file.

III. DECRYPTION PROCESS

A Conversion of Cipher Text into Predefined Format

The receiver read one character at a time from the cipher text file till he/she reach to the end of the file and converts the character into 8 bit binary format and store it into an array CIPHERTEXT with dimension 8.

B. Formation of Secret Value from Private-key for Decryption

Derive the secret value from the private-key by using step- C and store that 48-bit binary value into an array DERIVEDVALUE with dimension 48.

C. XOR operation and Formation of Decrypted Text

Step C.1 Perform XOR operation between the array CIPHERTEXT and DERIVEDVALUE. We consider 1st block, 2nd block , 3rd block, 4th block, 5th block and last block of the array DERIVEDVALUE for XOR operation in 1st pass, 2nd pass, 3rd pass, 4th pass, 5th pass, 6th pass respectively. Intermediate result is stored into an array IET with dimension 8. And final result is stored into array DECRYPTED with dimension 8. Corresponding ASCII code is generated from of the array DECRYPTED and from there we get the decrypted character which is stored into decrypted text file.

Example- we read a character '&' from the cipher text whose ASCII value is 38. We convert 38 in 8-bit binary representations and store it into an array CT. The secret value is derived in the step-C. Now we perform XOR operation between the cipher text and the secret value. Fig-9 represents the XOR procedure

Ciphertext1---Ciphertext8

0	**0**	**1**	**0**	**0**	**1**	**1**	**0**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV41---DV48

0	**1**	**0**	**0**	**1**	**1**	**1**	**0**
41th bit	42ndbit	43rd bit	44th bit	45th bit	46th bit	47th bit	48th bit

↓

Intermediate Decrypted Text (IDT (1))

0	**1**	**1**	**0**	**1**	**0**	**0**	**0**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV33---DV40

0	**1**	**0**	**1**	**0**	**1**	**0**	**1**
33th bit	34ndbit	35rd bit	36th bit	37th bit	38th bit	39th bit	40th bit

↓

Intermediate Decrypted Text (IDT (2))

0	**0**	1	1	1	**1**	**0**	**1**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV25---DV32

0	**1**	**1**	**0**	**0**	**0**	**1**	**0**
25th bit	26ndbit	27rd bit	28th bit	29th bit	30th bit	31th bit	32th bit

↓

Intermediate Decrypted Text (IDT (3))

0	1	0	**1**	**1**	**1**	**1**	**1**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV17---DV24

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
17th bit	18ndbit	19rd bit	20th bit	21th bit	22th bit	23rd bit	24rd bit

↓

Intermediate Decrypted Text (IDT (4))

0	**1**	**0**	**1**	**1**	**1**	**1**	**1**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV9---DV16

0	**0**	**0**	**0**	**0**	**0**	**0**	**0**
9th bit	10ndbit	11rd bit	12th bit	13th bit	14th bit	15th bit	16th bit

↓

Intermediate Decrypted Text (IDT (5))

0	**1**	**1**	**0**	**0**	**0**	**1**	**0**
1st bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

XOR operation

DV1---DV8

0	**0**	**1**	**1**	**1**	**1**	**0**	**1**
1th bit	2ndbit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

↓

www.ijitam.org

International Journal of Innovative Technology & Adaptive Management (IJITAM)
ISSN: 2347-3622, Volume-1, Issue-7 , April, 2014

Decrypted Text

0	1	1	0	0	0	1	0
1st bit	2nd bit	3rd bit	4th bit	5th bit	6th bit	7th bit	8th bit

Figure 9: Generation of Decrypted text by Block Wise Cumulative XOR operation between Cipher Text and Secret Value.

ASCII value corresponding to the decrypted text is 98 and the character corresponding to that ASCII is 'b'. In this way all the characters are decrypted and stored into the decrypted file and receiver is able to get the plain text.

IV. EXPERIMENTAL RESULT & DISCUSSION

The encryption of a plain text is done by using the 8th Palindrome number with a forward movement from the user-defined base value 10. The encryption or decryption has taken 31871 milliseconds. Table-I demonstrates the content of the source files, encrypted file and the decrypted file.

Table I: Corresponding content of source, encrypted, decrypted file

Content of the Source File (z.txt)	Content of the Encrypted File (en.txt)	Content of the Decrypted File (de.txt)
abcdefghijklmnopqrst uvwxyz	WTURSPQ^_\]Z[XYFG DEBC@ANOL	abcdefghijklmnopqr stuvwxyz

We have executed our program on 20 number of files of different types (*.com,*.txt,*.exe,*.sys and *.dll). The Execution results are being displayed in the Table II, Table III, Table IV and Table V, Table VI [5].

Table II: Execution Result for *.com files

Source File name	Source File size (Byte)	Encrypted File size (Byte)	Encryption / Decryption time (Mille seconds)
loadfix.com	1131	1131	32994
graphics.com	19694	19694	188457
diskcomp.com	9216	9216	123750
kb16.com	14710	14710	155442

Table III: Execution Result for *.txt files

Source File name	Source File size (Byte)	Encrypted File size (Byte)	Encryption / Decryption time (Mille seconds)
ReadMe.txt	286	286	28755
LICENSE.TXT	4829	4829	71099
TechNote.txt	9232	9232	107788
ROMAN.TXT	14423	14423	145287

Table IV: Execution Result for *.exe files

Source File name	Source File size (Byte)	Encrypted File size (Byte)	Encryption / Decryption time (Mille seconds)
cacls.exe	19968	19968	209737
mqsvc.exe	4608	4608	70793
label.exe	9728	9728	92089
shadow.exe	14848	14848	155814

Table V: Execution Result for *.sys files

Source File name	Source File size (Byte)	Encrypted File size (Byte)	Encryption / Decryption time (Mille seconds)
VIAPCI.SYS	2712	2712	51203
rootmdm.sys	5888	5888	90852
sffp_mmc.sys	10240	10240	109843
smclib.sys	14592	14592	152006

Table VI: Execution Result for *.DLL files

Source File name	Source File size (Byte)	Encrypted File size (Byte)	Encryption / Decryption time (Mille seconds)
MSMH.DLL	19768	19768	184782
KBDAL.DLL	6656	6656	84256
panmap.dll	10240	10240	135343
tcpmib.dll	14848	14848	157777

Figure 10 graphically shows how encryption time changes with size of the file being encrypted. It clearly indicates that the time required for encryption or decryptions do not depend on the type of the file, but depend only on its size.

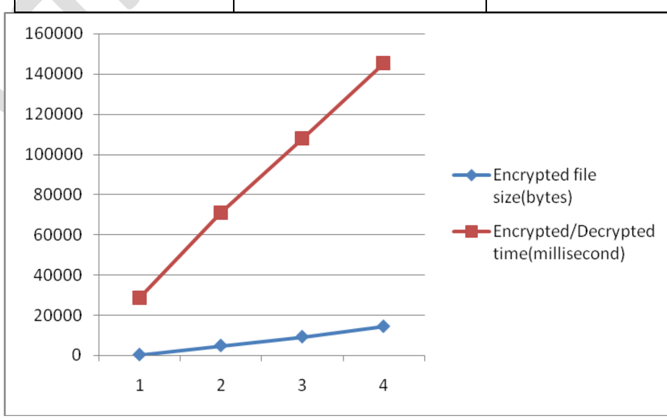

Figure 10: Relationship between Encryption Time and Source File Size

The Pearsonian Chi-square test has been performed here to decide whether the sample of encrypted files may be supposed to have arisen from a specified population. The Pearsonian Chi-square is defined as follows: $X2 = \Sigma \{(f\ 0 - f\ e)2 / f\ e\}$ Here f e and f 0 respectively stand for the frequency of a character in the source file and that of the same character in the corresponding encrypted file. [5]

If the Chi-square value is a higher one that means there is a higher frequency in source file and a lower frequency in encrypted file for a specific character. Incidentally it is observed in Table VII that the proposed encryption procedure having highly satisfactory chi-square values for all the files.

Table VII: Chi Square test Result on different files.

www.ijitam.org

Source File name	Source File size (Byte)	Encrypted File size (Byte)	Chi-square Test Value
diskcomp.com	9216	9216	251.0000
panmap.dll	10240	10240	5395.0000
mqsvc.exe	4608	4608	1075.0000
rootmdm.sys	5888	5888	452.0000
LICENSE.TXT	4829	4829	4743.0000

V. CONCLUSION

Here we proposed a private-key encryption scheme based on bitwise XOR operation between the plain text and the secret value. The secret value is the Nth palindrome number, counted by making a forward or backward movement from the user-defined base value. Where N is a positive integer in the range of (1<=N<=16).So as the base value or the Nth term is changed then the corresponding palindrome number is also being changed. Thus the security is increased.

The process of retrieving the secret value from the private-key is only being known by the receiver and the sender. So it is not possible for an unauthorised person to derive the secret value only with the presence of private-key. Thus the security is increased in a great entrance.

Besides this, a user can also do the encryption of an inputted file by using several numbers of distinct private-keys where each key is allotted for a specific block among several numbers of user-defined blocks in a plain text file. So the security is increased.

The size of the encrypted file is same as of the plain text file. So we don’t need any additional memory for encryption.

The execution time is depends on the file size not on the type of the file as we have done the encryption in bit level.

The only drawback is that if the value of the N or the base value is very higher then it will take very much time to generate the palindrome number. Thus the encryption or decryption time will be increased.

So, the proposed scheme is better in respect of providing security for encryption, encrypted file size management, encryption or decryption time requirement.

REFERENCES

[1] J. K. Mandal, S. Dutta, “A 256-bit recursive pair parity encoder for encryption ”, Advances D -2004, Vol. 9 nº1, Association for the Advancement of Modeling and Simulation Techniques in Enterprises (AMSE, France), www. AMSE-Modeling.org, pp. 1-14

[2] William Stallings, Cryptography and Network security: Principles and practice (Second Edition), Pearson Education Asia, Sixth Indian Reprint 2002.

[3] Atul Kahate (Manager, i-flex solution limited, Pune, India), Cryptography and Network security, Tata McGraw-Hill Publishing Company Limited.

[4] Mark Nelson, Jean-Loup Gailly, The Data Compression Book. BPB Publication

[5] Saurabh Dutta, “An Approach towards Development of Efficient Encryption Technique”, A thesis submitted to the University of North Bengal for the Degree of Ph.D., 2004.

www.ijitam.org