International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 4, April - 2013

An Approach Of Private Key Encryption Technique Based On Multiple Operators And Nth Even Or Odd Term For ASCII Value Of A Plain Text's Character.

Ramkrishna Das
Department of Computer Applications
Haldia Institute of Technology,
Haldia,WB,INDIA

Saurabh Dutta
Department of Computer Applications
Dr. B. C. Roy Engineering College
Durgapur-713206, WB, INDIA

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 person that needs it without it being intercepted.

Here, we have proposed an idea to increase the security of private key encryption technique. 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 making arithmetic operation between user defined base value and a derived decimal value. The decimal value is derived by taking the binary values from all 8-bit position of a plain text's character. The operators are supplied by the user. As the ASCII code is different for each distinct character, so we will get distinct secret private key value for each of the distinct character in the plain text. Thus the security is increased.

Keywords. *ASCII value, Nth Even or Odd term. Private key Encryption, Stream Cipher,*

1. Introduction.

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

Private Key cryptography refers to an encryption method where both the sender and the receiver share the same secret key or their keys are different, but they are related in an easily computable way. The Figure-1 represents a private key encryption method where the same secret key is being used for encryption and decryption.

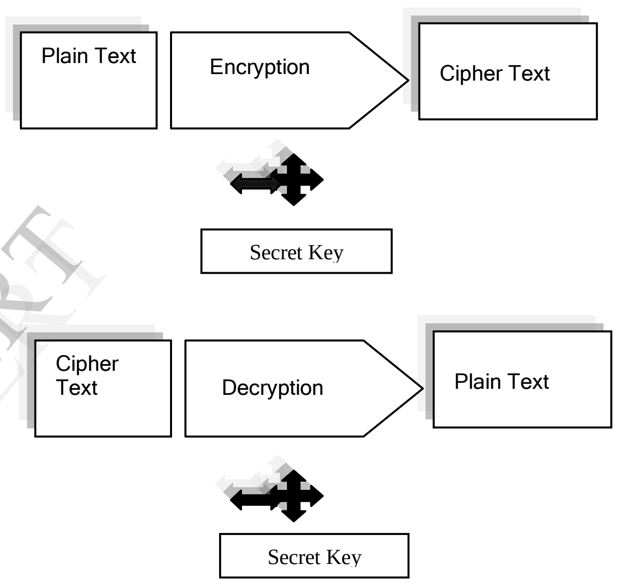


Figure 1: A private key encryption scheme

The main problem of private key encryption technique is to distribute the private key securely. All the existing private key encryption systems suffer from lack of security.

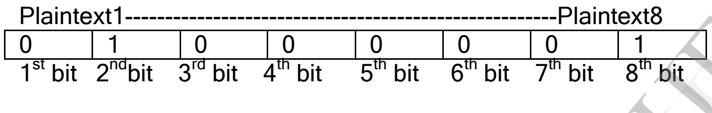
Here, we have proposed a scheme to increase the security of the existing private key encryption system. We have developed a secret procedure which is responsible to retrieve the secret value from the private key. The value is being used for encryption and decryption. The secret value is being generated by making different arithmetic operations (operations for odd values, operations for even values) between user defined base value and a derived decimal value. Those decimal values are being derived by taking the binary values from all bit positions of 8- bit representation of a plain text character. The operators are given by the user. As the ASCII code is different for each distinct character in plain text, so we must get distinct secret private key value for each of the distinct character in the plain text. Combination of user defined operators and derived secret value from even or odd bit position from plain text construct the private key. Hence the security is increased as we focus on securing the retrieving procedure rather than the private key value. Secret value can't be retrieved without the knowledge of the retrieving procedure. [2] [3] [4] [5].

In this paper section-2 describes the encryption process; section-3 describes the decryption process. An experimental result is being described in section-4 and section-5 draws the conclusion.

2. Encryption process.

Step-1: Plain text formation.

We take any type of file as input which is used as a plain text and stream cipher procedure is used for encryption. A unit of one character is being encrypted by the secret value at a time. Let 'A' is a character which is being present in the inputted file. ASCII value of 'A' is 65 so corresponding 8-bit binary representation is stored in the plain text. Fig-2 represents the 8 bit binary representation of a character 'A' from inputted file. The 8 bit binary representation is stored in an array PLAINTEXT [].


Plaintext1--Plaintext8

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

Figure 2: Formation of plain text

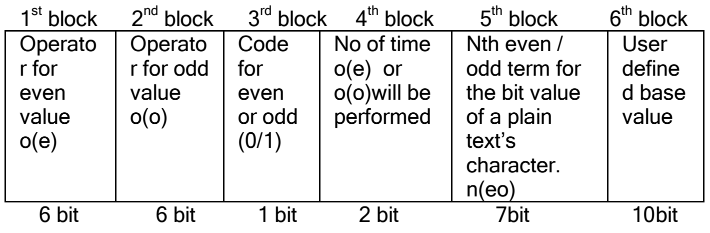
Step-1.1. Read the input file as plain text from the keyboard.

Step-1.2. Read one character at a time from the input file till we reached to the end of the file.

Step-1.3. Convert each character into 8-bit binary representation and store the value into an array PLAINTEXT [].

Step-2: key generation.

The size of the private key is 32 bits having 6 numbers of blocks. 1st block having size of 6 bits represent the operator for the even decimal values where the decimal value is derived from the even or odd position's binary value of the plain text.2nd block is having size of 6 bit represent the operator for odd value. 3rd block holds the value to determine that 5th block's value is even or odd. Size of the 3rd block is 1bit. 4th block holding the value which represents that how many times the even or odd operator is being executed. The size of the 4th block is 2 bits .5th block holding the value which represents the Nth even or odd term in the range of 0 to 2^7-1. Nth term is calculated from the bit value of single character from the plain text. The size of the 5th block is 7 bits. 6th block holds the user defined base value. The size of the 6th block is 10 bits. Fig-3 represents block diagram of the 32 bits Private key. [1][5]



1st block	2nd block	3rd block	4th block	5th block	6th block
Operator for even value o(e)	Operator for odd value o(o)	Code for even or odd (0/1)	No of time o(e) or o(o)will be performed	Nth even / odd term for the bit value of a plain text's character. n(eo)	User defined base value
6 bit	6 bit	1 bit	2 bit	7bit	10bit

Figure-3: Block diagram of 32 bit private key

Step 2.1. Read the user inputs for 1st, 2nd, 4th and 6th block from the user.

Step 2.2. The value for the 3rd, 5th will be calculated from the plain text.(inputted file).Convert the input values into corresponding bit size of their respective blocks and store the values in an array KEY [].

Step-3: Formation of secret value from private key for encryption.

One decimal value is being derived from the plain text. Decimal value (d(v)) is the value which is generated by taking the binary value of the 8 bit position of the plain text's character.

Step- 3.1. Determine the Nth even or odd term in the range of 0 to 2^7 -1 where Nth term is being calculated from the value of d(v).The d(v) value is the Nth odd or even value in the range of 0 to 2^7 -1 that value is stored in n(eo).

Step 3.2. The value (n (eo)) is an even or odd value. We perform a operation between n (eo) and the base value. 4th block's value in the private key determines that how many times the operation is being performed. The operation is selected (operator for even/operator for odd) depending on nature (even/odd) of the value n (eo).Thus we get the secret derived value.
.
Step 3.3. Make 8 bit binary representation of that final derived value and store that value in the array DERIVEDVALUE []

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.

-operator for even(+) -operator for odd (-) --even/odd----No. of time operation Will be done (2)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	1	0	1	1	1	0	1	1	0	1	1	1	0	0

------------------------------ ------------------------------ ----- ---------- --
6 bit 6 bit 1 bit 2 bit

-- Nth Even/odd term for-------------------base value (10) -----------
bit values of a
PT's character (33)

17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0

------------------------------ --
7 bit 10 bit

Figure 4: 32 bit representation of private key for specific value.

operator for even value, operator for odd value , base value and the value for number of time operation is performed are taken from user which are +,-,10 and 2 respectively. Convert this operators into corresponding ASCII codes and corresponding binary value is stored in 1st block,2nd block respectively. Binary representation of the base value is stored in block 6. 4th block hold the binary representation of the decimal value which represents how many times the operations is performed. We have read a character from the plain text. Let 'A' is the character. The ASCII code of 'A' is 65. Now we have converted that ASCII code into 8 bit binary representation. Which is 01000001. Now convert this binary value into decimal which is 65 and 65 is the 33th odd number in the range of (0 to 2^7 -1). So we perform 33-10=23; where 10 is the base value and '-' is the operator for the odd value. Now again 33 is subtracted with the result, that means ABS (23-33) =10.We have to perform the subtraction operation for two times as the 4th block's value in the private key is 2. If negative value can occur as a result then we take absolute value. We convert 10 into 8 bit binary representation and use it for encryption in sender end and for decryption in receiver end. The Secret value will be derived by using the private key. The Secret value will be stored in an array DV [8]. Fig-5 represents the entire procedure

1bit	2bit	3bit	4bit	5bit	6bi	7 bit	8bit
0	0	0	0	1	0	1	0

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

Figure 5: 8 bit representation of secret value derived from key.

Step-4: XOR operation and formation of cipher text file.

One block of the plain text is encrypted by the 1 block of the secret value cumulatively. XOR operation is performed bitwise between the plain text and the secret value. Fig-6 represents the encryption process

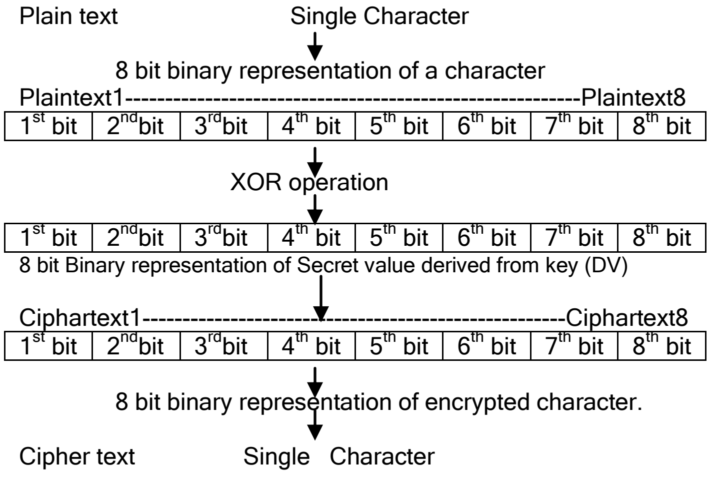

Plain text
Single Character
8 bit binary representation of a character
Plaintext1--Plaintext8
1st bit | 2nd bit | 3rd bit | 4th bit | 5th bit | 6th bit | 7th bit | 8th bit
XOR operation
1st bit | 2nd bit | 3rd bit | 4th bit | 5th bit | 6th bit | 7th bit | 8th bit
8 bit Binary representation of Secret value derived from key (DV)
Ciphertext1--Ciphertext8
1st bit | 2nd bit | 3rd bit | 4th bit | 5th bit | 6th bit | 7th bit | 8th bit
8 bit binary representation of encrypted character.
Cipher text
Single Character

Figure 6: XOR operation between plain text and secret value.

XOR operation is performed between plain text block and the block of the secret value derived from the private key (DV). Then we get the final encrypted block. Finally, we get the final encrypted single character from the encrypted block. In this way all the characters of the inputted file (plain text) are encrypted and the corresponding encrypted character are being generated. Those encrypted characters are stored into the cipher text file. The file is being sent to the receiver with the secret private key file. Fig-7 demonstrate the total XOR procedure between plain text and the secret value derived from the private key where PT is the plain text, DV is the secret value derived from the private key, and CT is the cipher text.

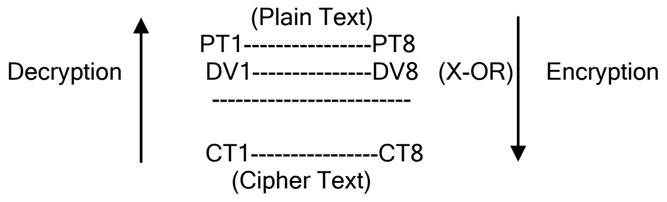


Figure 7: Block wise XOR operation between plain text and secret value derived from the private key.

Step 4.1. Perform XOR operation between PLAINTEXT [] and DERIVEDVALUE [1-8] and the final encrypted value will be stored in ENCRYPTED [].

Step 4.2. The binary value of the array ENCRYPTED [] is converted into corresponding ASCII value. Then the corresponding character is generated from the ASCII code and the character is stored into cipher text file.

Example- We have read a character 'A' from the inputted file and generate the plain text in Step-1 and secret value has been derived from the private key in the Step-3. Now we are going to perform the XOR operation between the plain text (PT []) and the secret value derived from the private key (DV []). Fig-8 represents the XOR procedure

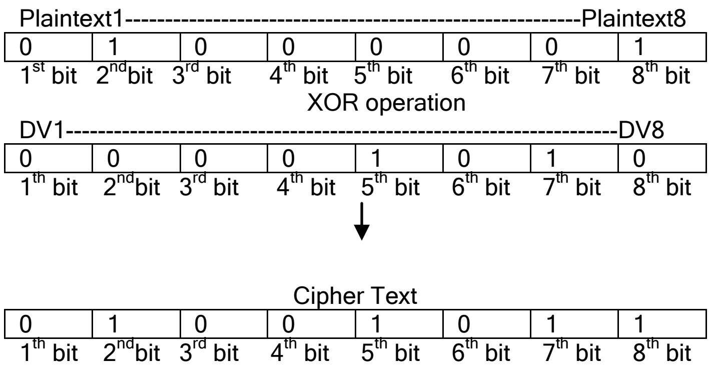

Plaintext1------Plaintext8
0 1 0 0 0 0 0 1
1st bit 2nd bit 3rd bit 4th bit 5th bit 6th bit 7th bit 8th bit
XOR operation
DV1------DV8
0 0 0 0 1 0 1 0
1th bit 2nd bit 3rd bit 4th bit 5th bit 6th bit 7th bit 8th bit
Cipher Text
0 1 0 0 1 0 1 1
1th bit 2nd bit 3rd bit 4th bit 5th bit 6th bit 7th bit 8th bit

Figure 8: Generation of cipher text by block wise XOR operation between plain text and secret value.

Corresponding ASCII value which is 75 generated from this 8 bit binary string (cipher text).Then corresponding character 'K' is generated from this ASCII value. In this way all the characters of the inputted file (plain text) are encrypted and the corresponding encrypted character are generated. Those characters are stored into the cipher text file. The encrypted file is sent to the receiver with the secret private key file.

3. Decryption process.

Step-5. Conversion of cipher text into predefined format.

The receiver get the cipher text file and the secret private key from the sender end. Read each of the encrypted character from the cipher text file and store it into 8 bit binary format into an array CIPHARTEXT [].

Step-6 Formation of secret value from private key for decryption.

Derive the secret value from the private key by using Step-3**.** and store the 8 bit binary value in the array DERIVEDVALUE [].

Step-7 XOR operation and formation of decrypted text file.

Step-7.1. Here CIPHERTEXT [] is used for decryption. Perform XOR operation between CIPHERTEXT [] and DERIVEDVALUE [1-8] and the final decrypted value will be stored in DECRYPTED [].

Step 7.2. The binary value of the array DECRYPTED [] is converted into corresponding ASCII value. The corresponding character is generated from the ASCII code and the character is stored into decrypted text file.

Example- We have read a character 'K' from the cipher text file. The ASCII value of 'K' is 75. We convert 75 in 8 bit binary representations and store it into CT []. The secret value has been derived from the private key in the Step-6. Now we are going to perform the XOR operation between the cipher text (CT []) and the secret value derived from the Private key (DV []). Fig-9 represents the XOR procedure

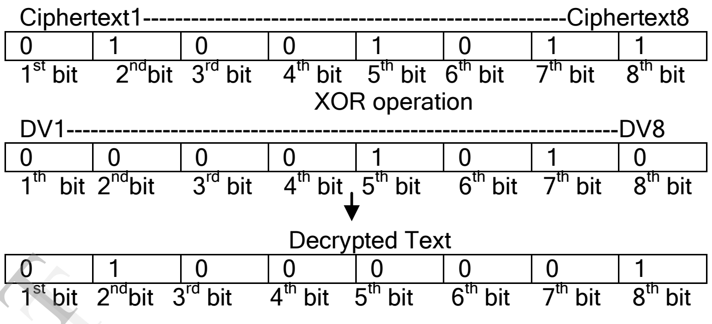

Ciphertext1------Ciphertext8
0 1 0 0 1 0 1 1
1st bit 2nd bit 3rd bit 4th bit 5th bit 6th bit 7th bit 8th bit
XOR operation
DV1------DV8
0 0 0 0 1 0 1 0
1th bit 2nd bit 3rd bit 4th bit 5th bit 6th bit 7th bit 8th bit
Decrypted Text
0 1 0 0 0 0 0 1
1st bit 2nd bit 3rd bit 4th bit 5th bit 6th bit 7th bit 8th bit

Figure 9: Generation of plain text by block wise XOR operation between cipher text and secret value derived from the key.

Corresponding ASCII value which is 65 will be generated from this 8 bit binary string (decrypted text).Then corresponding character which is 'A' is generated from this ASCII value. In this way all the characters of the cipher text file are decrypted. Those characters are stored into the decrypted text file and receiver is able to get the plain text.

4. Experimental result.

Here we have displayed the content of the plaintext, the corresponding encrypted content of the cipher text and the corresponding content of the decrypted file. Table-1 demonstrates the entire content of the source files, encrypted file and the decrypted file.

Table 1: corresponding content of source, encrypted, decrypted files.

Content of the source file (f.txt)	Content of the Encrypted file (f_en.txt)	Content of the Decrypted file (f_de.txt)
1234567890qweryy ASFGJKLZCBN [];',//!@#$%^&*()_+} {" :<>?	;9 ?v=z3 {}ossKY▯M-A▯ >I▯QW1- ▯%%+	1234567890qweryy ASFGJKLZCBN[]; ',//!@#$%^&*()_+} {" :<>?

	) /6▯-▯#U!wq -~zv5 -	

We use +,-,10,2 as operator for even value, operator for odd value, base value and no of time operation is performed respectively. The encryption is done by using the secret derived value from secret private key which is dedicated and may be distinct for each character of the plain text. The private key value is depending on the ASCII code of the character and the different operator inputted by the user. There will be N number of distinct private key value if a plain text has N number of characters. The encryption or decryption has taken 16328 milliseconds.
We have executed our program on 15 number of files of different types (*.com,*.txt,*.exe,*.sys and *.dll). We have taken 3 numbers of files of each type. The Execution results are being displayed in the following Tables.

Table 2:- 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	31703
README.COM	4217	4217	93312
diskcomp.com	9216	9216	313484

Table 3:- 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	204188
LICENSE.TXT	4829	4829	168640
TechNote.txt	9232	9232	314844

Table 4:- Execution result for *.exe files

Source File name	Source File size(Byte)	Encrypted file size (Byte)	Encryption/ decryption time(Mille seconds)
WINSTUB.EXE	578	578	46125
mqsvc.exe	4608	4608	75000
label.exe	9728	9728	326172

Table 5:- 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	128125
rootmdm.sys	5888	5888	210265
sffp_mmc.sys	10240	10240	354766

Table 6:- Execution result for *.DLL files

Source File name	Source File size (Byte)	Encrypted file size (Byte)	Encryption/ decryption time(Mille seconds)
iconlib.dll	2560	2560	164141
KBDAL.DLL	6656	6656	203125
panmap.dll	10240	10240	413844

If we are looking to the relation between encryption or decryption time and the source file size, then we get almost a straight line (Blue line) in figure 10.Which is a graphical representation of the relation between the encryptionor decryption time and the source file size. That means the relation is linier. As the source file size increase the encryption or decryption is increases and vice versa. The encryption or decryption time do not depend on the type of the file (.sys, .txt, .dll, .exe, .com). Fig-10 represents the relation between the encryption time or decryption time and the source file size.

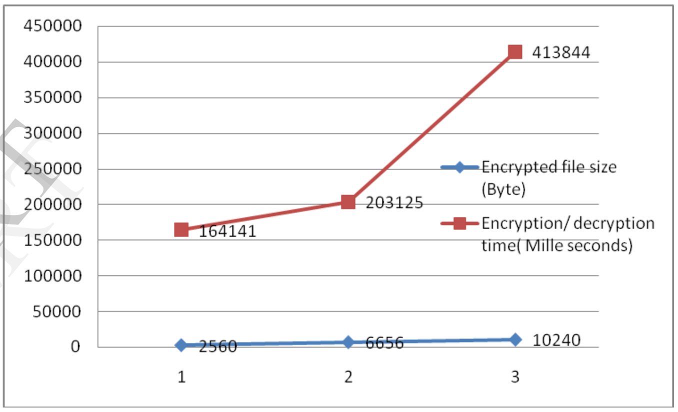


Figure 10: Relationship between encryption time and source file size

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 are 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. The "Pearsonian Chi-square" or the "Goodness-of-fit Chi-square" is defined as follows:
X2 = Σ {(ƒ 0 – ƒ e)2 / ƒ e}Here ƒ e and ƒ 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]
On the basis of this formula, the Chi-square values have been calculated for sample pairs of source and encrypted files. 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. So, this proposed encryption procedure generates highly secured encrypted file. Table 7 shows the Chi-square value for different types of file.

Table 7: Chi Square test Result on different files.

Source File name	Source File size (Byte)	Encrypted File size (Byte)	Chi square Test Value
loadfix.com	1131	1131	31305486336.00
iconlib.dll	2560	2560	18683279360.00
WINSTUB.EXE	578	578	16863553536.00
VIAPCI.SYS	2712	2712	15576862720.00
ReadMe.txt	286	286	276128064.00

5. Conclusion.

In this paper a new private key encryption scheme is proposed which is based on bitwise XOR operation between the plain text and the secret value derived from the private key.
The proposed method is simpler in term of mathematical computations compared to the existing private key encryption methods. The Secret value is calculated depending upon the ASCII code of a plain text's character and the different operators (base operator, operator for even value, operator for odd value) inputted by the user. There is separate secret key value for each of the character in the plain text as the ASCII code is different for each distinct character. Thus provide a great security.
The process of retrieving the secret value from the private key is only being known by the receiver and the sender. So if the private key value is available to a unauthorised person, still it is not possible to derived the secret value without the knowledge of secret value retrieving procedure from private key, thus the security is increased in a great entrance.
Besides this, a user can also do the encryption or decryption by using different types of operators in the private keys allotted for specific numbers of user defined blocks in a plain text file. So the security is increased as different keys are being used for encryption for different portion of plain text file,
The size of the encrypted or decrypted file is same as of the plain text file. So we don't need any additional memory for storing the encrypted or decrypted file.
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.Fig-10 representing the relation between the file size and the time graphically.
So, the proposed scheme is better in respect of providing security for encryption, encrypted file size management, encryption or decryption time requirement.

5. 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 EncryptionTechnique", A thesis submitted to the university of North Bengal for the Degree of Ph.D., 2004