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 and Prime Number In Selective Mode

Ramkrishna Das[1] ,Suman Sarkar[2], Kousik Paloi[3], Saurabh Dutta[4]

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 is 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 taking the Nth Vowel, Nth Consonant, Nth Semivowel and Nth Prime (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 **–** Nth Vowel, Nth Consonant, Nth Semivowel, Nth Prime, Stream Cipher private key encryption.

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.

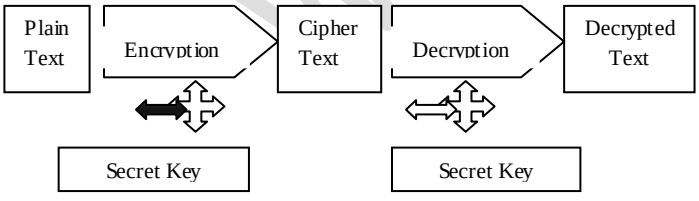


Figure 1: A Private-key Encryption Scheme

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. The secret value is generated by searching the Nth Vowel Nth Semivowel, Nth Consonant, again searching Nth Prime from a user-defined base value towards its forward or backward direction

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

Suman Sarkar[2] ,PG Student ,Department of Computer Science , Vidyasagar University, Paschim Medinipur , WB, INDIA, 9563805859, suman.sarkar_cs@yahoo.com.

Kousik Paloi[3] ,PG Student ,Department of Computer Science, Vidyasagar University, Paschim Medinipur, WB, INDIA, 9735757540, kousikpaloi@gmail.com

Saurabh Dutta[4], 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 prime number is a natural number which is greater than 1 and that has no positive divisors other than 1 and itself. For example, 5 is prime because 1 and 5 are its only integer factors. We have used the English alphabetic groups like vowel, semi-vowel, consonant 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].

A. Plain Text Formation

Let 'a' 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.

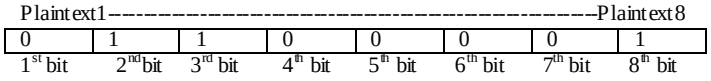


Figure 2: Formation of Plain Text

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.

B. key Generation

The size of the private-key is 40 bits having 7 blocks. 1st 5-bit block represents the choice of Sequence or combination number. 2nd 3-bit block represents the choice of Nth Vowel from English alphabet. 3rd 2-bit block holds the Nth Semivowel. 4th 5-bit block holds the Nth Consonant. 5th 2-bit block holds the Forward or Backward Movement.6th 6-bit block holds the Nth Prime. 7th 17-bit block holds the used defined base value.
Fig-3 represents block diagram of the 40-bit private-key.

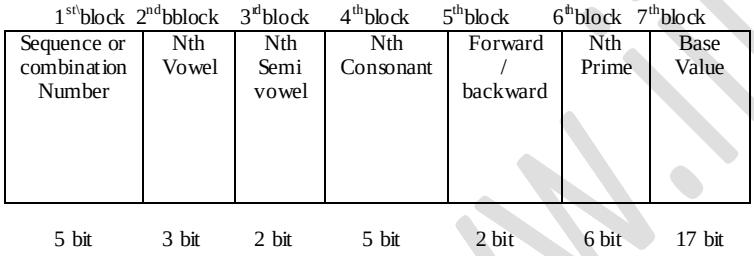


Figure-3: Block Diagram of 40-bit Private-key

Step B.1 Read the user inputs for 7 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

A user-defined sequence value is stored in the 1th block of the private-key The value of the 2nd block, 3rd block, 4th block are being used to determine the Nth term of vowel, semi-vowel, consonant respectively. Selection of Nth prime number is depending upon the 6st block's value. The value of the 5th block determines the forward or backward movement from the base value. The value of the 7th block holds the base value. Searching of Nth prime 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<=64).The user defined sequence value stored in 1st block is being used to combine the results (Nth (vowel, semi-vowel, consonant and prime number)) together.

Step C.2 Determine the Nth prime number by making a forward or backward movement from the user-defined base value and generate the corresponding Nth vowel ,semi-vowel, consonant.
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.

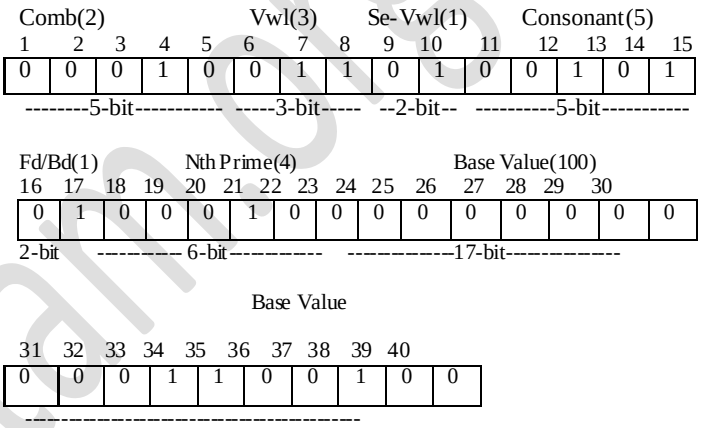
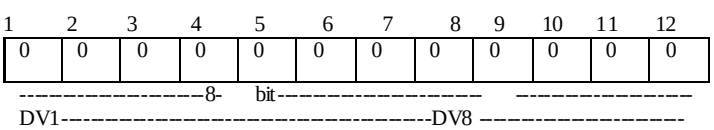


Figure 4: 40- bit Representation of Private-key for a Specific Value.

User-defined base value is 100 which is store in 7th block. 5th block of the private-key is 1 so the prime number is taken by making a forward movement. 2nd block of the private-key is 3, so the corresponding vowel is "i", 3rd block of the private-key is 1, so the corresponding semi-vowel is "w". 4th block of the private-key is 5, so the corresponding consonant is "g". As the 5th block value determines the forward movement so we have to make a forward movement (101,102,103...) from base value which is 100 .6th block represents the Nth term which is 4. So we take the 4th prime number with a forward movement from base value. The number is 109. We combine all this result as per the combination no which is 2.The combination number is store in 1st block. The definition of the combination is (Nth vowel + Nth semi-vowel + Nth prime + Nth consonant). The result is combining as per the combination value. The value is "iw109g".Convert the value into the ASCII code and generate binary code from their Fig-5 represents the 40-bit representation of secret value.



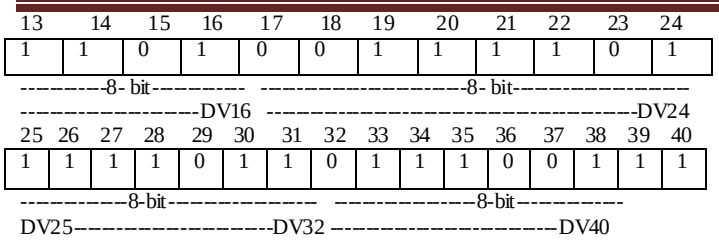


Figure 5: 40- bit Representation of Secret Value derived from Private-key.

D. XOR operation and Formation of Cipher Text

Plain text is being encrypted by the 5 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

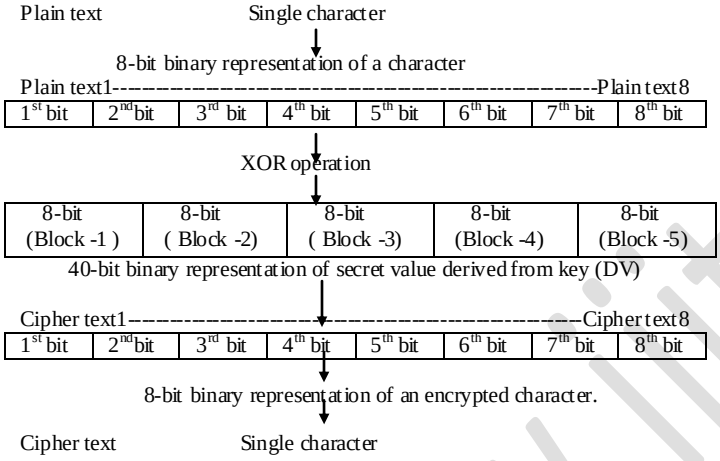
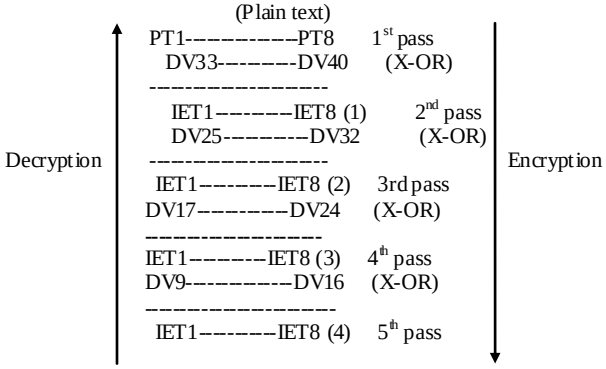


Figure 6: XOR operation between Plain Text and Secret Value.

Cumulative XOR operation is performed between plain text and the secret value.5th block,4th block,3rd block, 2nd block and the 1st block of the secret value (DV) are used for XOR respectively After 5 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



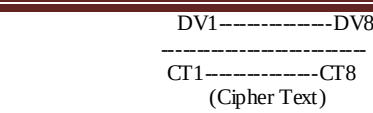


Figure 7: Block wise Cumulative XOR operation between Plain Text and Secret Value derived from the Private-key.

Step D.1 Perform XOR operation between the array PLAINTEXT and DERIVEDVALUE. We consider last block, middle block and 1st block of the array DERIVEDVALUE for XOR operation in 1st pass, 2nd pass and in 3rd 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 a' from plain text whose ASCII value is 97. 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

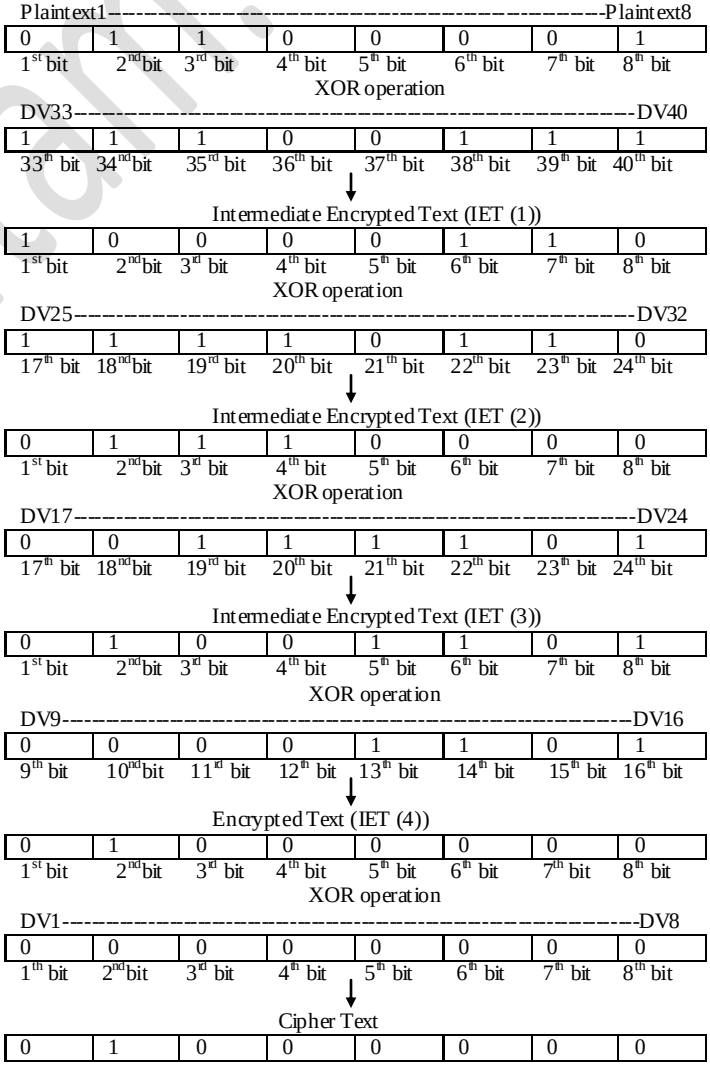


Figure 8: Generation of Cipher Text by Block Wise Cumulative XOR operation between Plain Text and Secret Value.

Intermediate

ASCII value corresponding to the cipher text is 64 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.

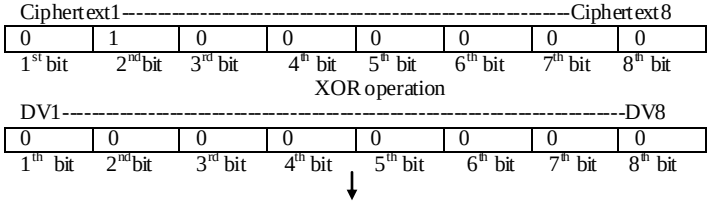
B. Formation of Secret Value from Private-key for Decryption

Derive the secret value from the private-key by using step- C and store that 40-bit binary value into an array DERIVEDVALUE with dimension 40.

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 and last block of the array DERIVEDVALUE for XOR operation in 1st pass, 2nd pass, 3rd pass, 4th pass,5th 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 64. We convert 64 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.



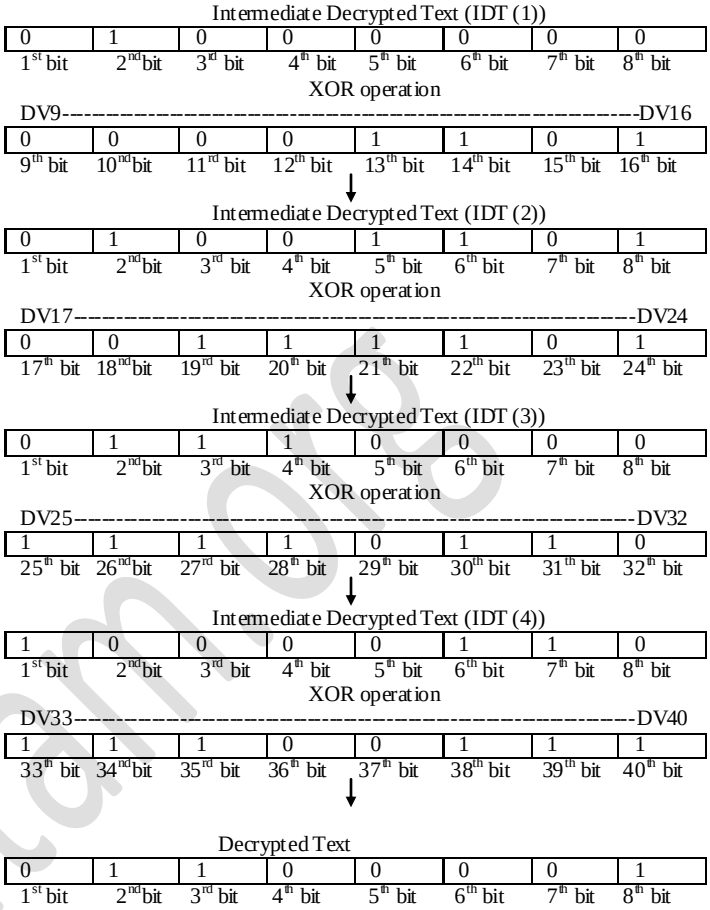


Figure 9: Generation of Decrypted text by Block Wise Cumulative XOR operation between Cipher Text and Secret Value.

ASCII value corresponding to the decrypted text is 97 and the character corresponding to that ASCII is 'a'. 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 4th prime number with a forward movement from the user-defined base value 100. The encryption or decryption has taken 18922 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 (f.txt)	Content of the Encrypted File (f_en.txt)	Content of the Decrypted File (f_de.txt)
qwertyuioplkjhgfdsa zxcbnm .	sugpv{wkmmihjedfqc xza`lo	qwertyuioplkjhgfdsazxc bnm .

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	52294
README.COM	4217	4217	127921
diskcomp.com	9216	9216	218038
kb16.com	14710	14710	273709

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	55975
LICENSE.TXT	4829	4829	155739
TechNote.txt	9232	9232	188882
ROMAN.TXT	14423	14423	270968

Table IV: 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	35883
mqsvc.exe	4608	4608	87773
label.exe	9728	9728	165797
shadow.exe	4848	4848	156108

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	83916
rootmdm.sys	5888	5888	115956
sffp_mmc.sys	10240	10240	226806
smclib.sys	14592	14592	225844

Table VI: 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	88577
KBDAL.DLL	6656	6656	172681
panmap.dll	10240	10240	171292
tcpmib.dll	14848	14848	347675

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.

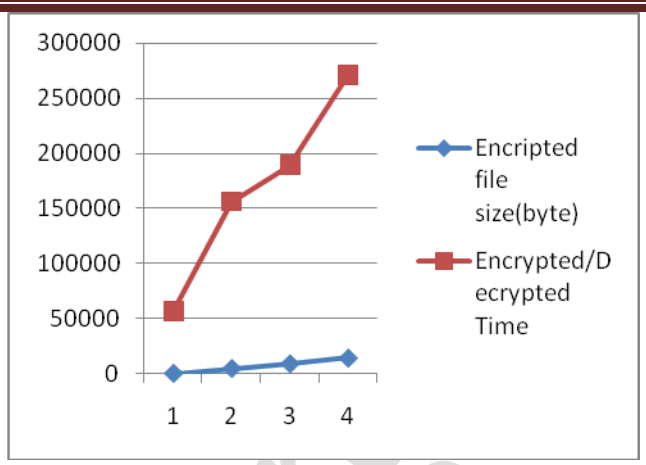


Figure 10: Relationship between Encryption Time and Source File Size

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.

Source File name	Source File size (Byte)	Encrypted File size (Byte)	Chi-square Test Value
kb16.com	14,710	14710	205.000000
KBDAL.DLL	6656	6656	375.000000
mqsvc.exe	4608	4608	1075.00000
Smclib. sys	14592	14592	5187.00000
AAB.TXT	20000	20000	500.000000

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 N^{th} prime number counted by making a forward or backward movement from the user-defined base value and combination of English alphabetic character. Where N is a positive integer in the range of (1<=N<=63).So as the base value or the Nth term is changed then the corresponding prime 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 prime Number and if the backward range of Prime number is very high from the base value then the Prime will not be found. Thus the encryption or decryption time will be increased or remained not encrypted.

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.