ISSN:0975-9646

Lakshmikanth Gurlanka et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014, 6404-6409

A Novel Secured Communication Channel Based On Genetic Functions

Lakshmikanth Gurlanka
M.Tech Student
Dept of CSE, AIET

Vasupalli Mahesh
Asst. Professor,
Dept of CSE, AIET

Y.Ramesh kumar
Asst. Professor,
Dept of CSE, AIET

Abstract: In this paper an encryption and decryption technique has been proposed and termed as A Genetic Functions Based Cryptosystem (GFC). Initiate's random numbers are generated with the help of Linear Method and Genetic Functions; CROSSOVER and MUTATION, and the first digit of each of these numbers are taken sequentially and stored in an array, termed as Collection Array. A block of characters are taken as input whose ASCII values are selected sequentially and stored in another array. Subtracting the numbers of Collection Array from the ASCII values does the encryption method. Applying a loop starting from ASCII value 0 to ASCII value 255 and from these values the cipher text is subtracted sequentially and comparing the result with the collection array do the decryption method and then the required ASCII value is chosen. A comparison of the proposed technique with existing and industrially accepted RSA,Triple-DES and AES has also been done in terms of encryption, decryption time; frequency distribution and non-homogeneity of source and encrypted files.

Keywords: encryption, decryption, AES algorithm, cryptosystem, genetic functions, mutation.

1. INTRODUCTION

Information security has become a very critical aspect of modern computing systems. With the global acceptance of the Internet, virtually every computer in the today is connected to every other. So at this point of time maintaining of secrecy and security of information has become necessity. For these reasons different types of research works on encryption and decryption is going on so that various algorithm are developed in this field. Encryption is a method that converts the plain text into non-readable one and Decryption is the method that converts the non-readable cipher text into readable plain text. Encryption is inversely proportional to Decryption.

Encryption and Decryption of data:

In cryptography,

Encryption: It is the process of encoding messages or information in such a way that only authorized parties can read it. In an encryption scheme, the message or information, referred to as plain-text, is encrypted using an encryption algorithm [11], turning it into an unreadable cipher text [6].

Decryption: It is the process of decoding the data which has been encrypted into a secret format. An authorized user can only decrypt data because decryption requires a secret key or password. In simple terms it is the conversion of cipher text into plain text [6].

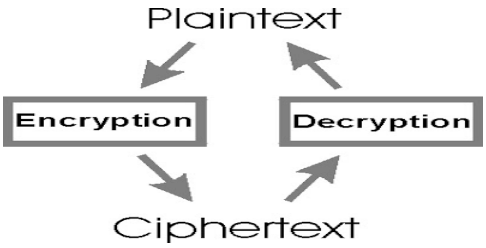


Figure 1: Conversion of plain text to cipher text and vice versa

This algorithm combines the features of Genetic Functions and Cryptography. Here we generate random numbers with the help of genetic functions "CROSSOVER" and "MUTATION". The algorithm contains a key of four parameters, for security.

Existing systems uses RSA algorithm or triple DES algorithms.

RSA: The RSA algorithm involves three steps: key generation, encryption and decryption.

Key generation:

RSA involves a public key and a private key. The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two distinct prime numbers p and q.
2. For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primitive test.
3. Compute $n = pq$,n is used as the modulus for both the public and private keys
   Compute $\varphi(n) = (p-1)(q-1)$, where φ is Euler's totient function.
   Choose an integer e such that $1 < e < \varphi(n)$ and greatest common divisor of $(e, \varphi(n)) = 1$; i.e., e and $\varphi(n)$ are coprime.
4. e is released as the public key exponent.
5. e having a short bit-length and small Hamming weight results in more efficient encryption - most commonly 0x10001 = 65,537. However, small values of e (such as have been shown to be less secure in some settings.[4]
6. Determine d as:
7. $$d \equiv e^{-1} \pmod{\varphi(n)}$$
8. i.e., d is the multiplicative inverse of e mod $\varphi(n)$.

This is more clearly stated as solve for d given (de) mod $\varphi(n) = 1$.This is often computed using the extended Euclidean algorithm. d is kept as the private key exponent.

The **public key** consists of the modulus *n* and the public (or encryption) exponent *e*. The **private key** consists of the modulus *n* and the private (or decryption) exponent *d* which must be kept secret.

Encryption: Alice transmits her public key (n, e) to Bob and keeps the private key secret. Bob then wishes to send message **M** to Alice.

He first turns **M** into an integer m, such that $0 < m < n$ by using an agreed-upon reversible protocol known as a padding scheme. He then computes the ciphertext c corresponding to

$$c = m^e \pmod{n}$$

This can be done quickly using the method of exponentiation by squaring. Bob then transmits c to Alice.

Note that at least nine values of m will yield a ciphertext c equal to m,[5] but this is very unlikely to occur in practice.

Decryption:

Alice can recover m from c by using her private key exponent d via computing

$$m = c^d \pmod{n}$$

Given m, she can recover the original message **M** by reversing the padding scheme.(In practice, there are more efficient methods of calculating c^d using the pre computed values below.)

It is vulnerable to the following attacks

- Timing attacks
- Adaptive chosen ciphertext attacks
- Side-channel analysis attacks

DES:DES (the Data Encryption Standard) is a symmetric block cipher developed by IBM. The algorithm uses a 56-bit key to encipher/decipher a 64-bit block of data. The key is always presented as a 64-bit block, every 8th bit of which is ignored. However, it is usual to set each 8th bit so that each group of 8 bits has an odd number of bits set to 1.

The algorithm is best suited to implementation in hardware, probably to discourage implementations in software, which tend to be slow by comparison. However, modern computers are so fast that satisfactory software implementations are readily available.

DES is the most widely used symmetric algorithm in the world, despite claims that the key length is too short. Ever since DES was first announced, controversy has raged about whether 56 bits is long enough to guarantee security.

The key length argument goes like this. Assuming that the only feasible attack on DES is to try each key in turn until the right one is found, then 1,000,000 machines each capable of testing 1,000,000 keys per second would find (on average) one key every 12 hours. Most reasonable people might find this rather comforting and a good measure of the strength of the algorithm.

Those who consider the exhaustive key-search attack to be a real possibility (and to be fair the technology to do such a search is becoming a reality) can overcome the problem by using double or triple length keys. In fact, double length keys have been recommended for the financial industry for many years.

Use of multiple length keys leads us to the Triple-DES algorithm, in which DES is applied three times. If we consider a triple length key to consist of three 56-bit keys K1, K2, K3 then encryption is as follows:

• Encrypt with K1
• Decrypt with K2
• Encrypt with K3

Decryption is the reverse process:

• Decrypt with K3
• Encrypt with K2
• Decrypt with K1

Setting K3 equal to K1 in these processes gives us a double length key K1, K2. Setting K1, K2 and K3 all equal to K has the same effect as using a single-length (56-bit key). Thus it is possible for a system using triple-DES to be compatible with a system using single-DES.

AES:

The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001[8]

AES is a block cipher, but it does not use a Feistel structure. The block size of AES is 128-bit, but the key size may differ as 128, 192, or 256 bits [9].

Substitution: This method substitutes each byte of the block in the order of S-box. It provides an invertible transformation of blocks during encryption, with the reverse during decryption.

Shifting Rows: This operation performs left circular shifts of rows 1, 2, and 3 by 1, 2 and 3,

Mix Columns: This method multiplies each column of the input block with a matrix. The multiplication operation is just like matrix multiplication, except that it uses a Finite Field to multiply two elements and performs an XOR operation instead of addition.

Add Rounded Keys: This operation just applies an XOR operation to each byte of the input block and the current weight (key) matrix.

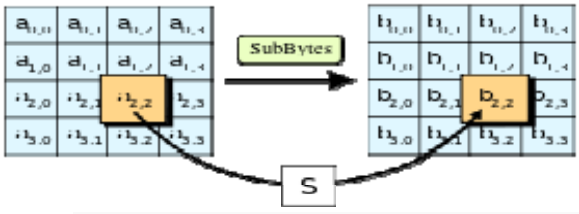


Figure2: the Sub-Bytes step, one of four stages in a round of AES

2. METHODOLOGY

The proposed algorithm consists of two steps i.e. random number generator and encryption

STEP-1 GENERATING RANDOM NUMBERS WITH THE HELP OF LINEAR METHOD AND GENETIC FUNCTIONS, i.e., CROSSOVER AND MUTATION RESPECTIVELY:

Assumption about the generation of random numbers: -

1. Representations of the numbers are in Binary.

2. Population Size is fixed to 10.
3. Hence 5 generations are required to generate 50 numbers if the number of block of characters is 50.

LINEAR METHOD:

The first generation is created with the help of linear method and its equation is given below: - The sequence of random numbers is obtained via the following iterative equation.

Xn+1= (a*Xn+c) mod m

Where 1. Xn is the seed value or it is the value of the first Chromosome of the first generation.
2. m = modulus (m > 0).
3. a = multiplier (0 <= a < m).
4. c=increment (0 <= c < m).

Once the numbers of the first generation is created the next generation numbers are generated using the GA operators CROSSOVER and MUTATION.

After generation 1, the numbers of the next generation is obtained by CROSSOVER followed by MUTATION. The pairing up of numbers is done first, with the concept that for odd type generation pairing is done in one way and for even type generation in the opposite way. For example, after the first generation we got the following numbers:-

333, 6578, 8614, 5959, 7922, 8837, 4440, 903, 3693,2686.

2nd Generation: - Pairing up: - (333, 6578), (8614, 5959), (7922, 8837),(4440, 903), (3693, 2686). For this generation crossover and mutation will take place let at 6th locus of the gene of chromosome.

CROSSOVER:

Binary Representation of the first pair:

333 = 000010**1001101**

6578 = 110011**0110010**

Crossover: 000010**0110010** 110011**1001101**

MUTATION:

Mutation: 00001**1**0110010 11001**0**1001101

= 434 =6577

Similarly, the other pairs can also be generated in the following way. Now after generating all the numbers by applying crossover and mutation on each pair we get;

434,6577,263,5798,8069,9202,4478,816,3646,2605

After the second generation we continue with the 3rd, 4th and 5th generation to generate 50 numbers (Each generation 10 populations) and get the final set of numbers.

STEP-2 ENCRYPTION:

1. Once all the numbers are generated then let this array of numbers be called SUB_ARRAY and select the first digit of each number from SUB_ARRAY and a new collection of numbers is generated and let this collection is called COLLECTION_ARRAY.
2. Use this numbers from COLLECTION_ARRAY sequentially for substituting on a one-to-one basis for the characters of the plain text

Use ASCII values of the plain text characters and subtract the numbers of COLLECTION_ARRAY from the ASCII values. For example the message "SOUMYA" the CIPHER TEXT will be calculated according to following method. LET SUB_ARRAY = {4167, 10117, 5602, 4867, 4307,2452}

Encryption:

Character	ASCII Value	Collection_Array Number Taken sequentially	Subtract	Result
S	83	4	83 - 4	79
O	79	1	79 - 1	78
U	85	5	85 - 5	80
M	77	4	77 - 4	73
Y	89	4	89 - 4	85
A	65	2	65 - 2	63

The enciphered message is "RESULT"

The Cipher text is: {79, 78, 80, 73, 85, 63}

Table1: encryption table

DECRYPTION:

Result (R)	For loop I =0 to 255 Do (I – R) & Compare With Collection_array & Choose I.	Charter
79	83	S
78	79	O
80	85	U
73	77	M
85	89	Y
63	65	A

Table2: decryption

3. DATA ANALYSIS

The ten text files of different sizes are taken for testing. The encryption time, the decryption time and source file sizes are noted for Triple-DES, RSA and GFC algorithms. Table 1 shows the encryption/decryption time of increasing size of text files for the proposed GFC, T-DES, and RSA technique. For any file size proposed GFCT takes less time to encrypt/decrypt compared to T-DES technique and takes more or less same time compared to RSA technique. From table it's seen that for the file **ctext.txt** the encryption time is 1 second whereas T-DES takes 49 seconds to encrypt the same file. For the same file RSA takes 1 second to encrypt. Hence it is seen that proposed GFC may be time efficient relative to RSA and T-DES in terms of text files. Figure 3 shows the pictorial representation of the same.

Source Filename (*.TXT)	Source File Size (Bytes)	Encryption Time (second)			Decryption Time (second)		
		RSA	T-DES	GFC	RSA	T-DES	GFC
adcajavas.txt	629	~ 0	~ 0	~ 0	~ 0	~ 0	~ 0
License.txt	7168	~ 0	2	~ 0	~ 0	2	~ 0
oladbjvs.txt	10240	~ 0	3	~ 0	~ 0	3	~ 0
NeroHistory	17408	~ 0	10	1	~ 0	10	1
Nero.txt	33792	~ 0	12	1	~ 0	12	1
whatsnew.txt	60633	1	26	1	1	27	1
new.txt	94208	1	35	1	1	35	1
ctext.txt	132096	1	49	1	1	50	1
redist.txt	540672	3	202	7	4	201	8
incidia.txt	1190912	6	431	18	7	430	18

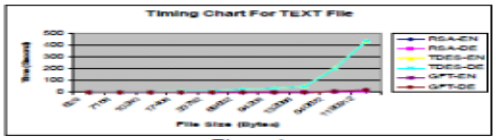


Figure 1
Encryption / Decryption Time for GFC, RSA, and T-DES techniques for TEXT files.

Figure3: text files comparison

File (*.EXE)	Source File Size (Bytes)	Encryption Time (second)			Decryption Time (second)		
		RSA	T-DES	GFC	RSA	T-DES	GFC
hypertrm.ex	28672	~ 0	13	~ 0	~ 0	14	~ 0
vlc.exe	96256	1	44	1	1	45	1
msimn.exe	130048	1	64	2	1	62	2
IEXPLORE	175104	1	83	2	2	85	3
wordpad.ex	292864	2	74	4	2	69	5
PINBALL.e	355328	3	83	5	3	82	5
dialer.exe	613376	4	151	11	5	150	12
ImageDrive	775168	5	190	15	6	174	16
winamp.exe	1307648	9	326	25	10	325	26
NeroStart Smart.exe	1835008	13	470	35	14	472	35

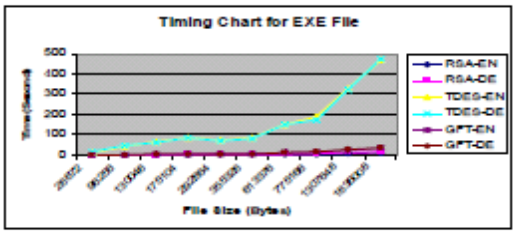


Figure 2
Encryption / Decryption Time for GFC, RSA, and T-DES techniques for EXE files.

Figure4: executable files comparison

Executable files:

The ten executable files of different sizes are taken for testing. Time taken for these files to encrypt / decrypt using proposed GFC is compared with the times taken for RSA and T-DES. Experiments results are given in Table 2. From Table 2 it is cleared that the proposed GFC technique takes less time to encrypt/decrypt compared to T-DES technique and takes more or less same time compared to RSA technique for any size of the executable files. The pictorial effects of the same are shown in Figure4

Studies on DLL Files

Time analysis has also been done for dynamic link libraries. Ten files of different sizes are taken for consideration. Table 3 shows the encryption and decryption taken for proposed GFC, T-DES and RSA techniques. It is seen from the table that the proposed GFC technique takes less time to encrypt/decrypt compared to T-DES technique and takes more or less same time compared to RSA technique for any size of the executable files. The pictorial effects of the same are shown in Figure 5

Analysis of Character Frequencies

Distribution of character frequencies are analyzed for text file for the proposed GFC, RSA and TDES algorithms. Figure 4 shows the pictorial representation of distribution of character frequencies for different techniques. Figure shows the distribution of characters in the source file "**redist.txt**". Figure b and c shows the distribution of characters in encrypted files both for RSA and T-DES respectively. Figure d gives the distribution of characters in encrypted file using the proposed technique GFC. It's seen from the picture that in the case of RSA the distribution of characters in encrypted file is concentrated in a small region, whereas both for TDES and the proposed technique GFC frequencies of encrypted file are distributed once the complete spectrum of characters. From this observation it may be conclude that the proposed technique may obtain good security

Table 3
File size v/s Encryption and Decryption Time for DLL files
(For GFC, RSA and T-DES algorithm)

Source Filename (*.DLL)	Source File Size (Bytes)	Encryption Time (second)			Decryption Time (second)		
		RSA	T-DES	GFC	RSA	T-DES	GFC
tataki.dll	65536	~ 0	17	1	1	18	1
wmpband.dll	98304	1	25	1	1	23	1
7zxa.dll	169984	1	41	1	1	41	1
wmpns.dll	221184	1	52	2	1	54	2
mpvis.dll	368640	3	87	3	3	87	4
Wmm2fxa.dll	502784	4	120	5	4	121	6
download.dll	655360	5	168	8	5	177	8
Rvsores.dll	753664	5	194	9	5	195	10
libeay32.dll	864256	7	225	10	7	224	12
bckgres.dll	1818624	13	493	25	13	493	27

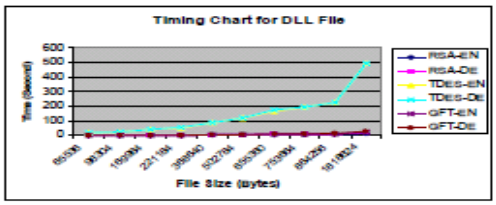


Figure 3
Encryption / Decryption Time for GFC, RSA, and T-DES techniques for DLL files

Figure 5

Tests for Non-Homogeneity

The well accepted parametric tests have been performed to test the non-homogeneity between source and encrypted files. The large Chi Square values may confirm the heterogeneity of the source and encrypted files. Text files are taken for experiment. The Chi Square test has been performed using source file and encrypted files for GFC technique and existing RSA and T-DES techniques. For non-homogeneity the value of the Chi Square should increase for the increasing file size. Five files of different sizes are taken. Further the high Chi Square value may ensure the non-homogeneity between source and encrypted files. In all three cases of implementation a good degree of non-homogeneity observed. So it may be inferred that proposed GFC technique may ensure optimal security in transmission. The pictorial representations of Chi Square values are given in figure 6.

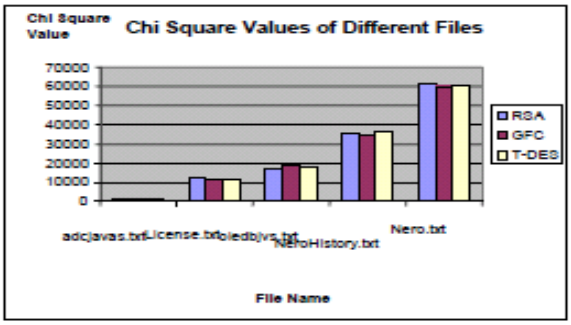
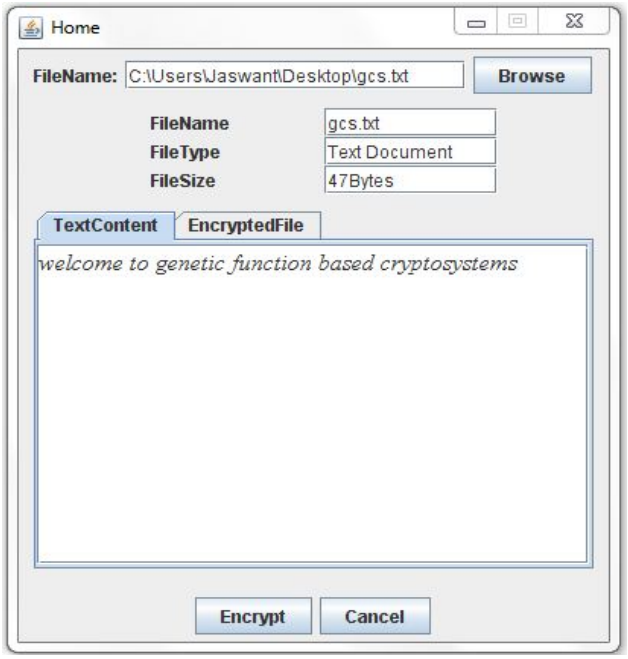


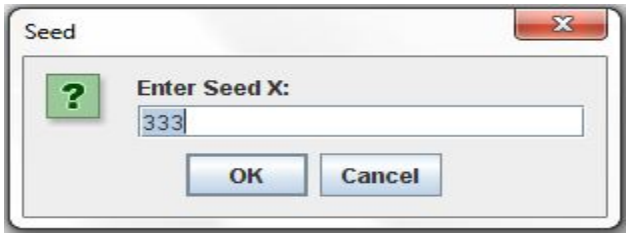
Figure 5
Chi Square Values for RSA, GFC and T-DES

Figure6: analysis

Home screen:



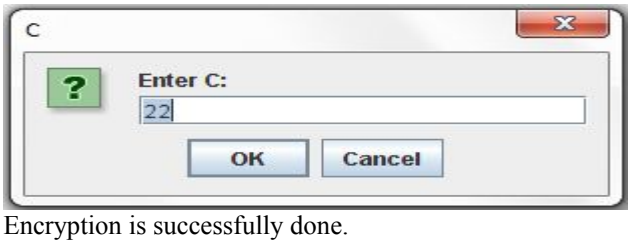
SEED VALUE:



MOD VALUE:

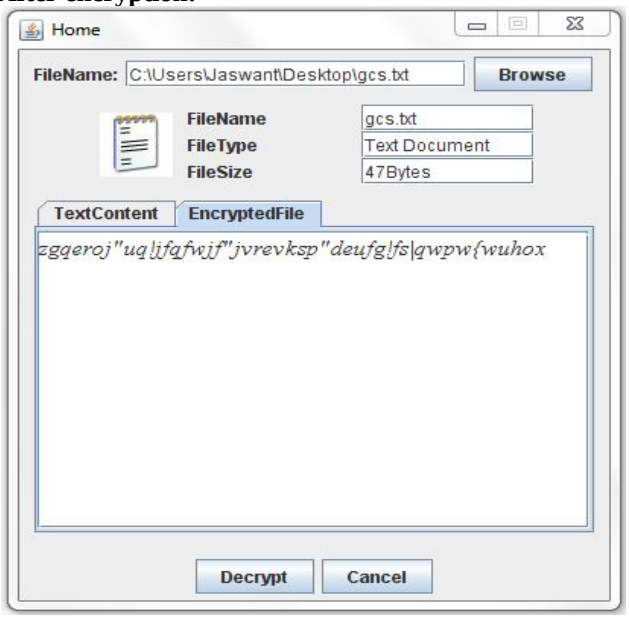
MULTIPLIER VALUE:

CARRY VALUE:

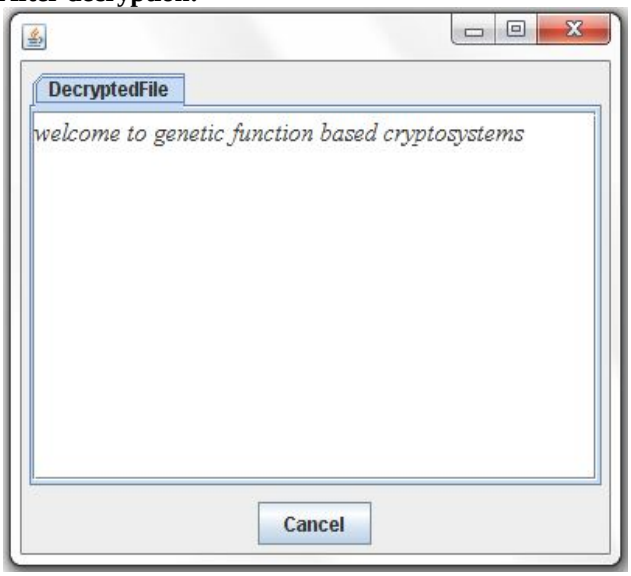


Encryption is successfully done.

After encryption:



After decryption:



4. CONCLUSION

The proposed technique "GENETIC FUNCTION TECHNIQUE" presented in this paper is simple and easy to implement cryptographic system. The generation of random numbers with the help of genetic functions provides a unique feature in this technique. This random numbers are used to encrypt the original message and as this numbers are randomly generated it is very tough to break the cipher text. The proposed technique may appear to produce a computationally non-breakable cipher text. The result of the Frequency-Distribution tests shows the fact that the cipher characters are distributed wide enough, and it is also seen that the source and the encrypted files are non-homogenous which is established by Chi Square tests. It produces a competitive Chi Square value while comparing with the RSA system.

REFERENCES:

[1] Atul Kahate, "Cryptography and Network Security",Tata McGraw-Hill, 2nd Edition.

[2] Byron S. Gottfried, "Programming with C" TATA McGraw HILL, Second Edition, 1998.

[3] Herbert Schildt, "Java: The Complete Reference ", Tata McGraw-Hill Publishing Company Limited, 5th Edition.

[4] E.Balagurusamy, "Programming with Java", Tata McGraw-Hill Publishing Company Limited, 3rd Edition

[5] Ankit Fadia, "Network Security", Macmillan India Ltd.

[6] William Stallings, "Cryptography and Network Security", Prentice Hall, 3rd Edition.

[7] Subhranil Som, Joytsna Kumar Mandal, (2009) "Random Byte Value Shift (RBVS) Algorithm", JIS Management Vista, Vol. III, No. 1, pp.81-88.

[8] Som S., Mitra D., Halder J., (2008) "Session Key Based Manipulated Iteration Encryption Technique (SKBMIET)", The 2008 International Conference On Advanced Computer Theory And Engineering ICACTE 2008), 20-22, December 2008, Phuket, Thailand.

[9] Som S., Bhattacharyya K.., Roy Guha R., Mandal J.K.,(2009) "Block Wise Bits Manipulations Technique (BBMT)", The 2009 International Conference On Advanced Computing, 6-8, August 2009, Tiruchirappalli, India.

[10] Som S., Mandal J. K., (2008) "A Session Key BasedSecure-Bit Encryption Technique (SBET)", NationalConference (INDIACom-2008) on Computing ForNation Development, February 08-09, 2008, New Delhi, India.

[11] Som S., Mitra D., Halder J., (2008) "Secure-Bit Rotate and Swapped Encryption Technique (SBRSET)", National Conference on Trend in Modern Engineering System (IConTiMES 2008), February 23-24, 2008, WB, India