A Genetic Functions Based Cryptosystem (GFC)

Subhranil Som[1], Joytsna Kumar Mandal[2] and Soumya Basu[3]

[1]Department of Computer Application, JIS College of Engineering, Kalyani
[2]Department of Computer Science & Engineering, University of Kalyani
[3]Swasti Nursing Home, West Bengal

Summary

In this paper an encryption and decryption technique has been proposed and termed as A Genetic Functions Based Cryptosystem (GFC). Initiates 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 and Triple-DES has also been done in terms of encryption, decryption time; frequency distribution and non-homogeneity of source and encrypted files.

Keywords:

Genetic Functions Cryptographic Technique (GFC), Linear Method, Genetic Functions, Encryption, Decryption.

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 [1]. So at this point of time maintaining of secrecy and security of information has become necessity [5]. 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.

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.

KEY= {Seed Value (Xn), Modulus (m), Multiplier (a), Incrementer(c)}

Where,
Xn, a, c, m is the parameters of the linear method, which are known to only sender and receiver.

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

Section 2 of the paper discussed about the proposed scheme with example. Section 3 shows the results of some implementation on different files with analysis about the proposed technique. Section 4 deals with discussion and conclusions and references are drawn in Section 5.

2. The scheme

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.

Manuscript received September 5, 2009
Manuscript revised September 20, 2009

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:
000011**1**0110010 110010**1**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.
3. 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}**

STEP-3 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

3. Results and analysis

In this section the implementation of different types of files are presented. The text, executable and dynamic link libraries files are taken for experiments. This implementation has been done using high-level language.

3.1 Studies on Text Files

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 1 shows the pictorial representation of the same.

Table 1
File size v/s Encryption and Decryption Time for TXT files
(For GFC, RSA and T-DES algorithm)

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
oledbjvs.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.tx	69632	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

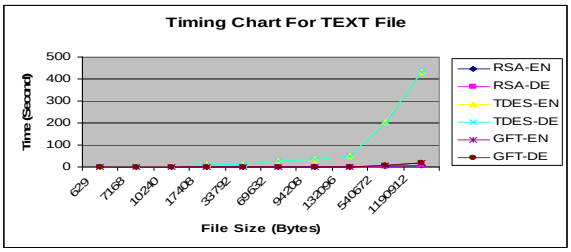

Figure 1
Encryption / Decryption Time for GFC, RSA, and T-DES techniques for TEXT files.

3.2 Studies on 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 Figure 2.

Table 2
File size v/s Encryption and Decryption Time for EXE files
(For GFC, RSA and T-DES algorithm)

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.exe	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

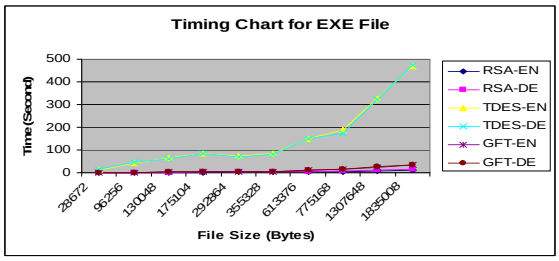

Figure 2
Encryption / Decryption Time for GFC, RSA, and T-DES techniques for EXE files.

3.3 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 3.

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
Rvseres.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

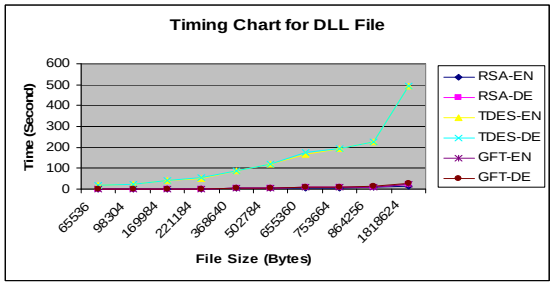

Figure 3
Encryption / Decryption Time for GFC, RSA, and T-DES techniques for DLL files.

3.4 Analysis of Character Frequencies

Distribution of character frequencies are analyzed for text file for the proposed GFC, RSA and T-DES algorithms. Figure 4 shows the pictorial representation of distribution of character frequencies for different techniques. Figure a 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 T-DES 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.

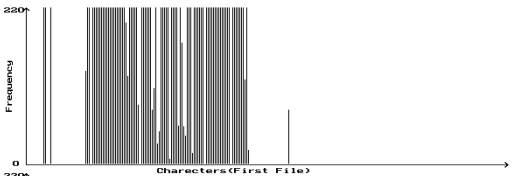

Figure a: Distribution of characters in the source file "**redist.txt**"

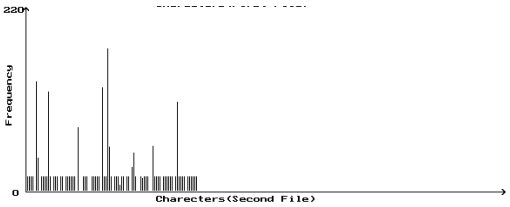

Figure b: Distribution of characters in the encrypted file of RSA

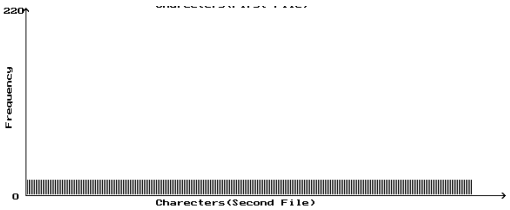

Figure c: Distribution of characters in the encrypted file of T-DES

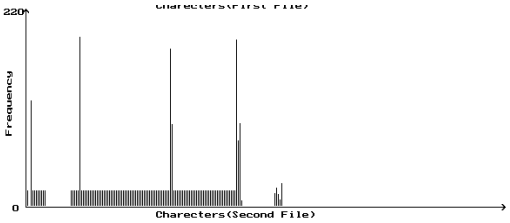

Figure d: Distribution of characters in the encrypted file of GFC

Figure 4
Frequency distribution graph of the file "redist.txt" for RSA, T-DES, and GFC as the source file

3.5 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 5.

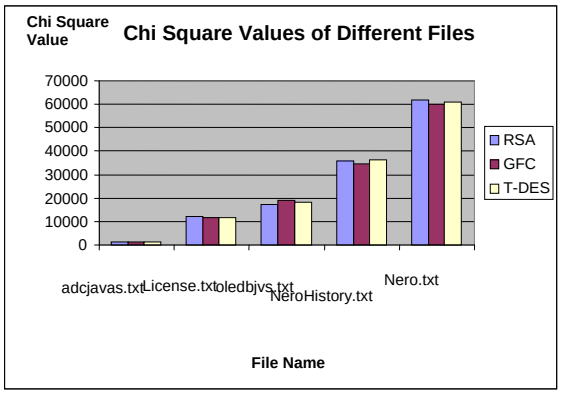

Figure 5
Chi Square Values for RSA, GFC and T-DES

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

5. Future scope

The future scope of this project is to include more features of Genetic Algorithms in this newly developed cryptographic algorithm to make it more secure and effective. In this project with the help of genetic functions we are generating random numbers but in future we can calculate the fitness of each of these numbers and can be more selective as the random numbers, which are generated, will be selected according to their fitness level.

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 Based Secure-Bit Encryption Technique (SBET)", National Conference (INDIACom-2008) on Computing For Nation 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.

Mr. Subhranil Som received his Master degree in Computer Application in 2003. Currently he is pursuing his PhD in Technology from University of Kalyani, West Bengal, India. He holds a distinction in Physics and Math in Graduation. His fields of interest include Cryptography and Network Security, Java, C. He is currently Lecturer in Computer Application Department, JIS College of Engineering, West Bengal India. He was attached with a WHO's International Research Project on "e-Health for Health Care Delivery", University of New South Wales, Sydney, Australia. He has finished several courses related to computer Application, object oriented analysis and design, Software Engineering and Project Management.

Dr. JYOTSNA KUMAR MANDAL received his M.Tech. and PhD degree from Calcutta University. He is currently Professor of Computer Science & Engineering & Dean, Faculty of Engineering, Technology & Management, University of Kalyani, West Bengal India. He is attached with several AICTE projects. He has 21 years Teaching & Research Experiences.

Mr. Soumya Basu received his Graduation and Master degree in Computer Application from West Bengal University of Technology. He is currently attcahed Swasti Nursing Home in Systems, West Bengal, India.