Cryptosystem for Providing Secured Application based on Genetic Algorithm

Amanie Alhussain
Information Technology Department
Peoples' Friendship University
Moscow, Russia
amanie-alhussain@hotmail.com

Prof.Vadim Stefanyuk
Information Technology Department
Peoples' Friendship University
Moscow, Russia
stefanukv@iitp.ru

Abstract— In this study, has shown how to design asymmetric encryption algorithm based on using two pseudorandom number generators and the features of crossover and mutation operations of Genetic algorithm (GA).the roles of pseudorandom number generators, in the proposed algorithm, help to generate dynamic representation for each character and define where to apply the crossover and mutation operations in deterministic way to produce the ciphertext;

The proposed encryption algorithm solves the problem of exchange and distribution of private keys over the networks by using crossover operator;

The various examples and implementation of the algorithm proves that it exchanges the keys and encrypts successfully all the characters, and serves the goals of cryptography.

Keywords: Genetic Algorithm, Mutation, Crossover, public key, Cryptography , Deterministic Approach, Asymmetric Key Encryption , Pseudorandom Number Generators

1.INTRODUCTION

Cryptography plays an important role in the network security. Cryptography is the science of writing in secret code. The purpose of cryptography is to protect transmitted information from being read and understood by anyone except the intended recipient; so this paper has designed Asymmetric cryptosystem based on the genetic algorithm operations to implement and achieve this purpose;

Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using different keys—one is a public key and the other is a private one. It is also known as public-key encryption. [4,5].Asymmetric encryption transforms plaintext into ciphertext using one of the two keys ,and the plaintext is recovered from the ciphertext using the other paired one.[4]
The concept of public-key cryptography evolved from an attempt to attack one of the most difficult problems associated with symmetric encryption. The problem is that of key distribution, which requires either (1) that two communicants already share a key, which somehow has been distributed to them; or (2) the use of a key distribution center[4].

Up to Genetic algorithm, it is a search heuristics that mimics the process of natural evolution. Genetic algorithms are based on the Darwinian theory of evolution. Genetic algorithms have been invented by J. Holland in 1960s [6], and since then they have been successfully applied to the variety of problems in the field of combinatorial optimization [7].

In literature to date, Many genetic algorithms based encryption have been proposed. A. Tragha et al.[1, 2], describe a new symmetrical block ciphering system named ICIGA (Improved Cryptography Inspired by Genetic Algorithms) which generate a session key in a random process. The block sizes and the key lengths are variable and can be fixed by the user at the beginning of ciphering. ICIGA is an enhancement of the system (GIC) "Genetic algorithms Inspired Cryptography" [3].

In this paper we propose an approach for asymmetric encryption and key exchange. First we generate pseudorandom number sequences using Linear Congruently and Lagged Fibonacci generator and from them calculate new sequence of numbers; each of these sequence plays a different role in the proposed encryption algorithm; the algorithm would generate internally dynamic representation for each character , which is depend on the private key.

The rest of the paper is organized as follows: In Section 2, we introduce the proposed Pseudorandom Number Generators; Section3 discusses about the proposed encryption algorithm; Section 4 experimental results; Section 5 concludes the paper.

2. PSEUDORANDOM NUMBER GENERATORS

2.1. A pseudorandom number generator (PRNG), also known as a deterministic random bit generator (DRBG), is an algorithm for generating a sequence of numbers that approximates the properties of random numbers.

2.2. Classes that used to produce PRNG:

- ✓ **Lagged Fibonacci generator**

The Fibonacci sequence may be described by the recurrence relation: $\mathbf{X_{n+1}= (X_n+X_{n-1})\ mod\ m}$ where **m** module, $\mathbf{X_0}$ initial value.

- ✓ **Linear Congruential generator**

The generator is defined by the recurrence relation:
$\mathbf{X_{n+1}= (a\ X_n\ +c)\ mod\ m}$ Where **X** is the sequence of pseudorandom values, and **m** –"modulus" **a**– the "multiplier"; **c**- the "increment", $\mathbf{X_0}$ – the "seed "or "initial value".

FREE

2.3. Roles of PRNG in the proposed encryption algorithm

The encryption algorithm uses three sequences of numbers, which each of them plays different role in the algorithm. (As shown in Fig.1); the sequences are:

a) Linear Congruential generator, which is used to define where to mutate.
b) Lagged Fibonacci generator, which is used to define where to crossover.
c) Generated Sequence of numbers, which is created internally and related to both Linear Congruential and Lagged Fibonacci generator; it is defined by the relation:
$((X_n+X_{n-1}) \bmod m + (aX_n+c) \bmod m))\bmod m$
And it is used to create dynamic representation for each character;

Example of PRNG and Generated Sequence with the parameters (X_0=7, m=100, a=6, c=9, size=100)is shown in Fig.2 and its implementation in Fig9.

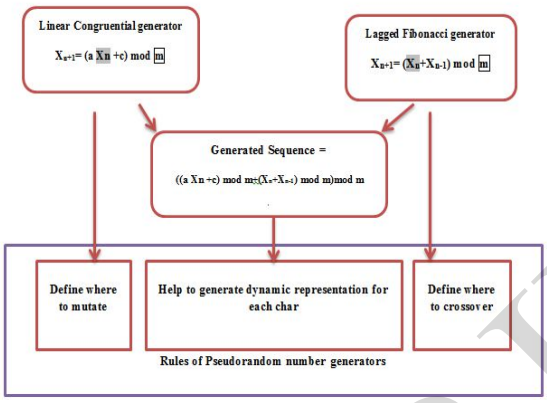


Fig.1 Schema of the Roles of the three sequences of numbers, and the Shared parameters among them.

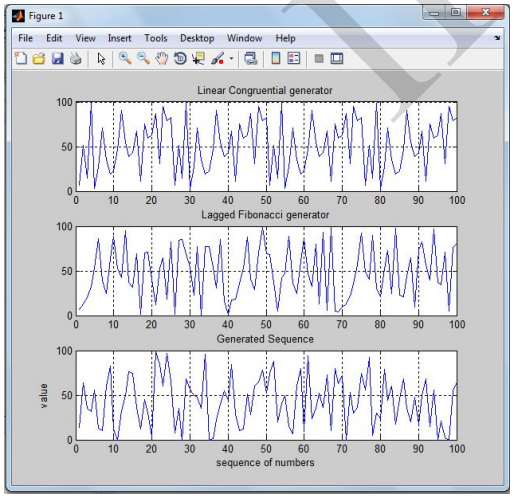


Fig.2 Example of the three Sequences of numbers with predefined parameters.

3. THE PROPOSED ENCRYPTION ALGORITHM:

3.1. The Key Generation

3.1.1. Private Key

In the key generation procedure, five parameters are used in private key which provide strength to the algorithm rendering it difficult for cryptanalysis by intruder. The five parameters of the private key are: ***Private Key= {initial value, modulus, multiplier, increment, size}***

initial value , modulus, multiplier , increment are the parameters of Linear Congruential generator; but it would be used also to produce Lagged Fibonacci generator, and Generated Sequence of numbers which mentioned in section 2.3 .

size is the buffer in which random numbers are generated. *(Fig2. Shows the different values of the three sequences of numbers, when the same private key is used which share the same parameters among the sequences)*

3.1.2. Public key

3.1.2.1. Decryption of the Algorithm:

The steps to generate public key could be summarized as follow:

1) Generate random number (R_n) in the range 0-15, and convert it into hexadecimal value $(HEX\ (R_n))_{16}$.
2) Convert each component of private key into hexadecimal value $(HEX\ (PK_n))_{16}$.
3) XOR between hexadecimal representations of both random number and private key components:
$(HEX\ (XOR_n))_{16} = (HEX\ (R_n))_{16} \oplus (HEX\ (PK_n))_{16}$.
4) Convert the values obtained in step3 into binary (8bit) representation $(HEX\ (XOR_n))_2$ and add the binary (8bit) representation of random number $(HEX\ (R_n))_2$ to the end of numbers.
5) Crossover the two consecutive bytes of the data obtained in step4 according to the middle of its size i.e.4bits (here the crossover type is one point crossover);
6) Convert the produced bytes in step 5 into Hexadecimal values $(HEX\ (CrossoverBytes))_{16}$.
7) Public key = sequence of the produced hexadecimal numbers in step6.

3.1.2.2. Example of applying the public key algorithm:

step1: *Generate random number (Rn) in the range 0-15:*
$Rn= (12)_{10}=(C)_{16}$;

step2: *Convert each component of private key into hexadecimal value:*
Initial value$=(7)_{10}=(7)_{16}$, modulus$=(100)_{10}=(64)_{16}$,

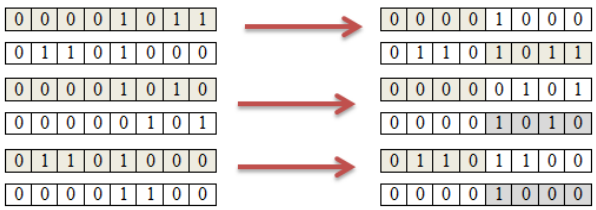
IRJECE

multiplier=$(6)_{10}$=$(6)_{16}$,Increment=$(9)_{10}$=$(9)_{16}$,
size=$(100)_{10}$=$(64)_{16}$

step3: *XOR between hexadecimal representations of both random number and private key components:*
$(C)_{16}\oplus(7)_{16}$ =$(B)_{16}$; $(C)_{16}\oplus(64)_{16}$ =$(68)_{16}$; $(C)_{16}\oplus(6)_{16}$ =$(A)_{16}$;
$(C)_{16}\oplus(9)_{16}$ =$(5)_{16}$; $(C)_{16}\oplus(64)_{16}$ =$(68)_{16}$;

step4: *Convert the values obtained in step3 into binary (8bit) representation and add the binary (8bit) representation of random number to the end of numbers.*
$(B)_{16}$=$(0000\ 1011)_2$; $(68)_{16}$ =$(0110\ 1000)_2$; $(A)_{16}$=$(0000\ 1010)_2$;
$(5)_{16}$ =$(\ 0000\ 0101)_2$; $(68)_{16}$ =$(0110\ 1000)_2$; $(C)_{16}$=$(0000\ 1100)_2$

step5: *Crossover the two consecutive bytes of the data obtained in step4 according to the middle of its size:*



step6: *Convert the produced bytes in step 5 into Hexadecimal values*
$(00001000)_2$=$(8)_{16}$;$(01101011)_2$=$(6B)_{16}$;
$(00000101)_2$=$(5)_{16}$;$(00001010)_2$=$(A)_{16}$;
$(0110\ 1100)_2$=$(6C)_{16}$; ;$(\ 0000\ 1000)_2$=$(8)_{16}$

step7: Public key = sequence of the produced hexadecimal numbers in step6.
Public key=8 6B 5 A 6C 8
(Note: that the public key for the same private key is not static, but it depends on the random generated number).

3.1.2.3. Role of the public key in the proposed encryption algorithm:

The public key would be sent directly from the sender to the receiver , before start the encryption process (Fig.3) , so the receiver would retrieve the private key from the public key by reverse the steps of obtaining the public key and based on the symmetry of the operations involved and the symmetry of XOR operations.

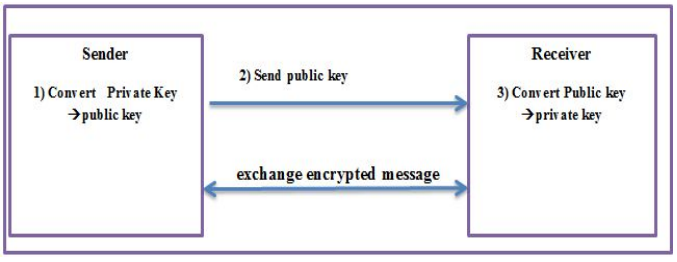


Fig3. The role of public key in the proposed encryption algorithm.

By sending the public key before starting the encryption process, we solve the problem of private key storing and distributions;

3.2. Encryption Process

Before starting the encryption; there is a step of preparing and sending the public key to the receiver (this step is done just one time), after that a set of characters with corresponding values should be generated based on the addition of the value of ASCII cod to Generated Sequence of numbers (which is created based on both the linear and Fibonacci pseudorandom numbers); let's called it as "characters table".

Attention should be taken that each character in characters table must be unique (more conditions and constrains implemented inside cryptosystem).the benefit here is that the ASCII values would not be static, it would vary depending on the secret key (this is increase one more level of security);

Encryption Steps:

1. Get the value of the first character in plaintext from characters table, and Convert its value into binary (8bits)
2. Apply XOR operation between the obtained value in step1, and the binary representation (8bits) of Generated Sequence of numbers.
3. Take mode 8 of the Linear Congruential and Fibonacci sequence to get decimal values ranging from 0 to7 Ln =mod (Ln, 8) , Fn =mod (Fn, 8)
4. Apply mutation based on the mod 8 of Linear Congruential Ln =mod (Ln, 8), in the values obtained in step 2.
5. Crossover the two consecutive bytes of the data stream obtained in step4 according to the mod 8 of Fibonacci Sequence Fn =mod (Fn, 8)

3.3. Decryption Process:

Before starting the decryption, if it is the first time, a private key should be retrieved from public key as descried earlier, then the three sequences of numbers should be generated, after that "characters table" should be created in the

IRJECE

same way mentioned in encryption, finally the steps of decryption are just reversal of the encryption.

4. EXPERIMENTAL RESULTS:

4.1. Exchange the keys

Before starting the encryption process, the generated public key based on the used private key should be sent firstly as mentioned in section *3.1.2*; this give the cryptosystem the ability to change the pair of keys without caring about the problem of key distributions and storing;

4.1.1. On the Sender side:

1) Enter the private key that would be used in encryption.(as shown in Fig.4)

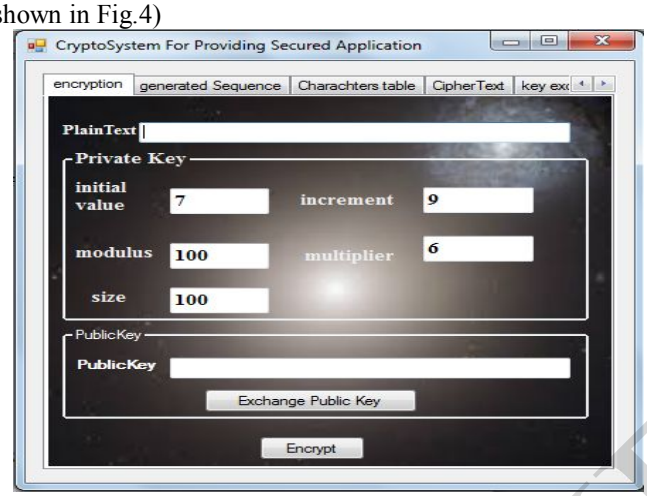


Fig4. The private key on the sender side.

2) Press the button " Exchange Public key"

The public key would be calculated on sender side (as shown in Fig5.) ,and send directly to receiver side (shown in Fig6);

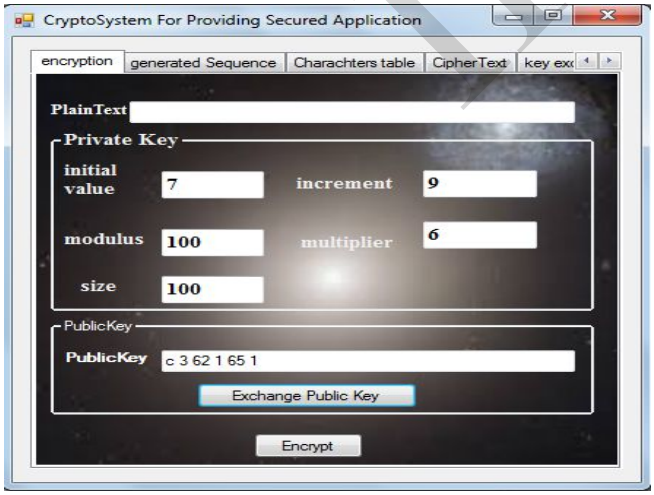


Fig5. Calculation of public key on sender side

4.1.2. Receiver side:

After the public key is received (shown in Fig6.), the receiver could calculate the private key from the received public key when pressing the button "retrieve private key", (shown in Fig7).

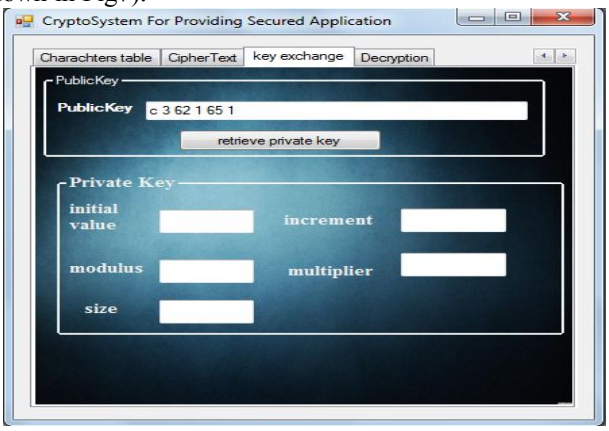


Fig6. The received public key on the receiver side.

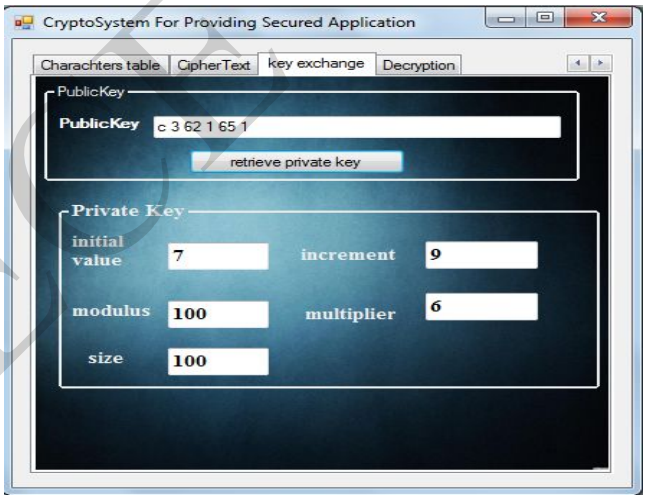


Fig7. Calculation of the private key from the received public key on the receiver side

4.2. Encryption

Plaintext: "cryptosystem"

If the parameters of private key {initial value=7, modulus=100, multiplier =6, increment=9, size=100} (as shown in Fig8) is chosen; the Linear Congruential ,Fibonacci and Generated Sequence of numbers which is created using the private key is shown in Fig9 (also Fig.2);

And the dynamic generated "characters table" which is based on the private key is shown in Fig.10.

The ciphertext would be=" ab fb a7 b6 e7 9b 71 b9 d7 28 83 87 "as shown in Fig11 ;

In decryption process we will obtain the

plaintext= "cryptosystem" as shown in Fig12;

IRJECE

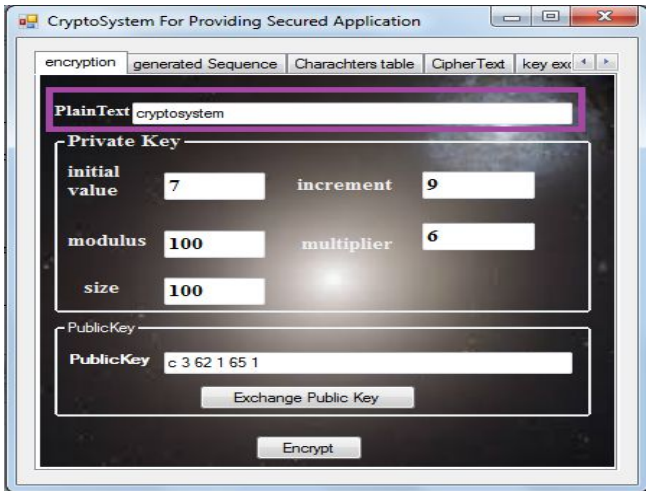


Fig.8 plain text and private key

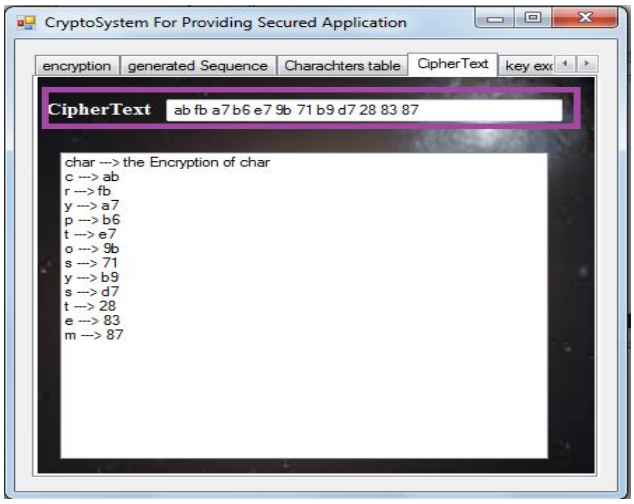


Fig.11 ciphertext.

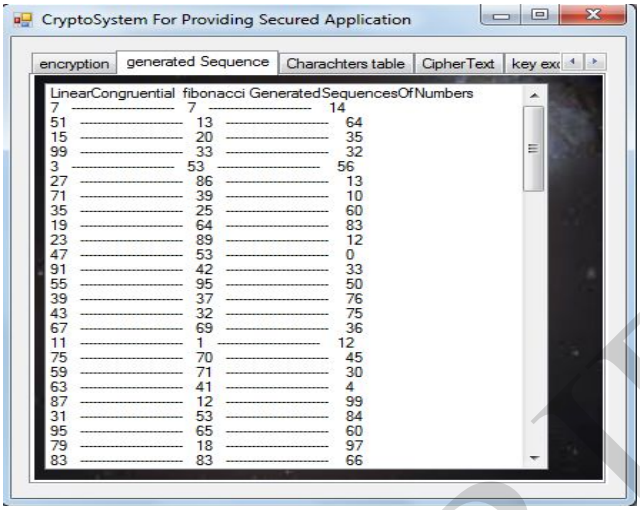


Fig.9 the generated sequences of numbers (Linear, Fibonacci and Generated Sequence)

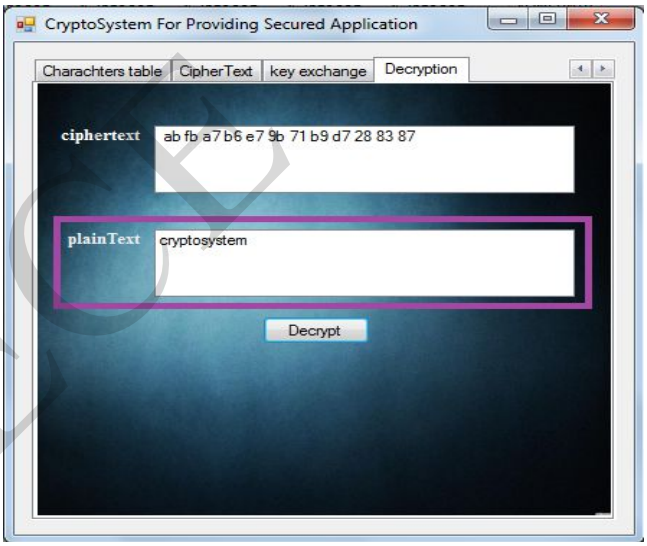


Fig12. The produced plaintext on receiver side

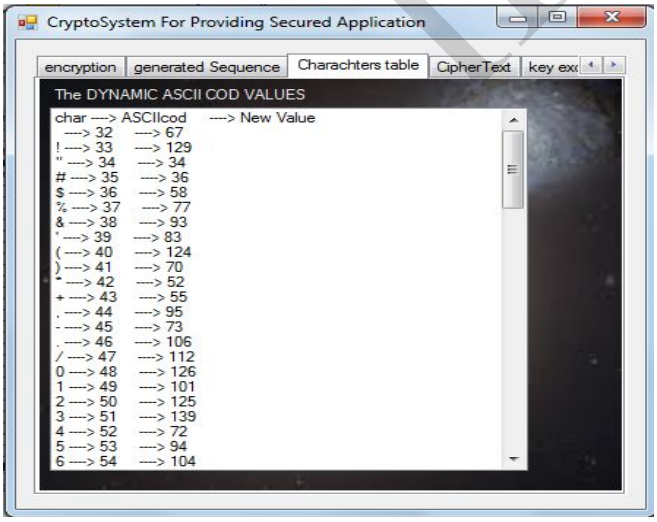


Fig.10 the generated characters table

5. CONCLUSIONS

In the present study a cryptographic algorithm has been designed using the concept of genetic algorithms with the randomness properties of PRNG. The cryptographic algorithm serves two goals, the first one is the encryption and the second one is the key distribution and storing; both of these targets use the genetic algorithm operators.

The work has been implemented using C#; and MATLAB is used as simulation platform. It has been tested in all character of English, Russian and Arabic and work successfully.

This algorithm enhances the quality, efficiency and effectiveness of the algorithm being used for the cryptography.

IRJECE

6. ACKNOWLEDGMENT

I would like to articulate my deep gratitude to my Prof. Vadim Stefanyuk who has always been my motivation for carrying out the research, Information Technology Department, at peoples' Friendship University in Moscow where the computational works have been carried out.

7. REFERENCES

[1] Menzes A. J., Paul, C., Van Dorschot, V., Vanstone, S. A., "Handbook of Applied Cryptography", CRS Press 5th Printing; 2001.
[2] National Bureau Standards, "Data Encryption Standard (DES)," FIPS Publication 46; 1977.
[3]Tragha A., Omary F., Mouloudi A.,"ICIGA:Improved Cryptography Inspired by Genetic Algorithms", Proceedings of the International Conference on Hybrid Information Technology (ICHIT'06), pp. 335-341, 2006.
[4] Cryptography And Network Security Principles And Practice Fifth Edition William Stallings ISBN 13: 978-0-13-609704-4
[5]Cryptography and Data Security Edited by: Dorothy Elizabeth Rob, ling Denning PURDUE UNIVERSITY ISBN 0-201-10150-5
[6]Holland, J., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, 1975.
[7]Poonam Garg, "Genetic Algorithm, Tabu Search & Simulated Annealing Attack on Transposition Cipher", proceeding of third AIMS International conference on management at IIMA – 2006, 983-989

IRJECE