Optimized Symmetric Keys Generated using Genetic Algorithm for Fully Homomorphic Encryption System

K.Kalaiselvi, S. Gopika and Mary Jacob
Faculty, Department of Computer Science, Kristu Jayanti College (Autonomous), Bangalore, India.

ABSTRACT

With growing demands in the computing services and increase in computation power, the computer consumers are migrating towards cloud computing domain to increase the effectiveness in compute, share, manage and to store the resources. The data storage and computations are delegated to the third party to manage it in the cloud that poses breach of confidentiality during computation. The cloud providers should ensure security for the data in rest and data in motion to prevent data disclosure during computation. Data encryption is a best way to secure the data, but practically not feasible to share the secret keys since third party is involved in handling the data. A solution to this problem is to perform the computations without decrypting the cipher text. Fully Homomorphic Encryption (FHE) is an efficient method to provide random computations on the encrypted data stored in cloud without decrypting it. The computation results are also encrypted which can be decrypted from the user domain. Genetic Algorithm (GA) is a heuristic search method based on natural selection. This paper proposes a symmetric key homomorphic encryption based on Genetic Algorithm key generation. Due to the randomness in key generation a strong cryptosystem is designed that makes the Chosen plain text attack more difficult. The results shows that the keys generated by GA are more random than other symmetric key cryptosystems.

KEY WORDS: FULLY HOMOMORPHIC ENCRYPTION (FHE), GENETIC ALGORITHM (GA), SYMMETRIC KEYS, CLOUD COMPUTING.

INTRODUCTION

Cloud computing is the new buzz word in the domain of computation, though mostly associated with faster computing solutions and pay-use model, there is always a threat to the data which is stored and handled in the cloud. The data which is accessed is no longer in the control of the cloud client. Data encryption is considered as a vital form of protecting the data against breach of confidentiality and security (William et al., 2011). Conventional cryptographic methods use One Time Password or Pseudo Random Number Generators to generate the cryptographic keys for encrypting and decrypting the data. Encryption schemes are needed to be provided to all data in rest, motion and in use.

Other mechanisms like SSL can be used to provide security for data in motion. But the real challenge lies in protecting the data in rest and in use (Bensitel et al., 2016). Though the computations are delegated to a third party in the cloud, the public or private keys for the cryptographic operations cannot be shared with the cloud providers. So, a practically possible Homomorphic schemes are needed that permit the user to perform computation on the ciphertext, access the encrypted data records, to validate the computations performed by the third party. Homomorphism is a mathematical property in which preserves the mapping between two algebraic structures belonging to the same type. Fully homomorphic encryption (FHE) enables the user to accomplish the computations on the ciphertext without decrypting the data (Gentry et al., 2009).

Biosc Biotech Res Comm P-ISSN: 0974-6455 E-ISSN: 2321-4007

Identifiers and Pagination	**Article Information**
Year: 2021 Vol: 14 No (6) Special Issue	Received: 15th April 2021
Pages: 339-343	Accepted after revision: 04th June 2021

This is an open access article under Creative Commons License Attribn 4.0 Intl (CC-BY).

DOI: *http://dx.doi.org/10.21786/bbrc/14.6.70*

The result of the computation is in the encrypted form which can be decrypted from the client domain to generates the same output that is generated by the unencrypted original data. The privacy and security of the data stored in cloud is preserved by FHE schemes since data access and computations can be performed in the encrypted data without decrypting them. Genetic Algorithm (GA) belongs to the Evolutionary Algorithms, that is best suitable for increasing the complexity using randomness (Goldberg et al., 1989) The encryption key needed for the cryptographic process can be generated randomly by applying GA that strengthens the encryption process. This paper proposes a novel method for protecting the data stored in cloud using AES-FHE scheme. The symmetric keys needed for the encryption are generated using GA and AES-FHE with modified rounds is proposed in this paper. The paper is organized as follows: Section 1.1 and 1.2 explains the basic concepts of Genetic Algorithm and Fully Homomorphic Encryption, Section 2discusses the proposed work, Section 3 provides the Experimental result analysis, Section 4 presents the conclusion and future work.

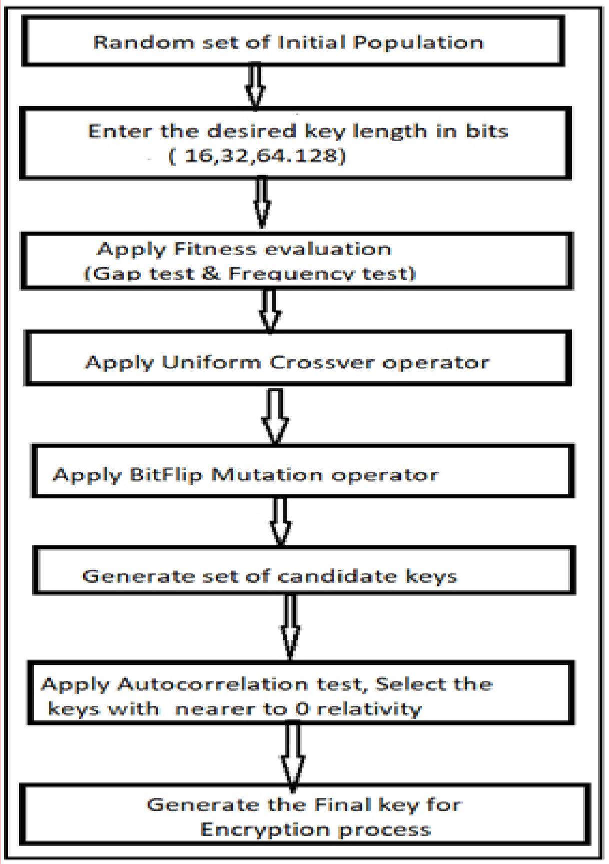
1.1 Genetic Algorithm: This paper explores the Genetic Algorithm to generate the distinct and highly randomized key for cryptographic operations that makes the encryption process more strong and secure. GA is well proven Evolutionary computation method inspired from Darwin's natural selection through survival of fittest process. GA follows the natural Genetics principles, mimics the natural reproduction process and produce a strong new population. Using this natural property, its already proven that GA can generate strong cryptographic keys for both symmetric and Asymmetric algorithms (Sindhuja et al. 2014) GA initiates its search for the candidate solution, known as chromosomes in the problem space from initial random population and iteratively performs the search to reach the feasible solution with three basic operators.

Selection operator selects the chromosomes to generate the next population. Crossover operator broadcasts the strong set of chromosomes with high fitness value by mating the individual chromosomes. Mutation operator increases the diverse combination and prevents the search from being stuck in local best solution. The strength of the GA depends on the parameters used to setup the experimental environments such as Encoding methods, types of operators used and control parameters (Kalaiselvi et al., 2017) Control parameters includes the number of generations run and the population size. The generic binary coding, uniform crossover, Roulette Wheel Selection and Inverse Mutation are used to develop the algorithm which generates the unique key for the encryption process for the AES based FHE scheme. The pseudo code for a simple genetic algorithm can be described as follows.

```
Begin
Generate the initial population randomly.
Calculate the fitness value.
Repeat
Roulette Wheel Selection: a pair of parents based on fitness value.

Uniform Crossover: Create two offspring.
Bit FlipMutation: Apply to each child.
Evaluate: Fitness percentage.
New population: All the offspring are new.
Until (Terminate condition = TRUE).
End.
```

1.2 Fully Homomorphic Algorithm: Homomorphism in mathematic property describes the algebraic function that maps any two algebraic structures that belongs to the same groups or rings or vector spaces. Using this property Homomorphic Encryption (HE) enables the cloud providers to perform computation on the ciphertext. In conventional encryption methods, the users need to negotiate with the data security in order to use the cloud services. To preserve confidentiality and integrity of the data in cloud HE allows the operations on encrypted data. Depending on the computational process and the repeated times usage it can be Partially Homomorphic (PHE), Somewhat Homomorphic Encryption (SWHE) or Fully Homomorphic Encryption (FHE) (Van Dijk et al, 2010 and Yookesh et al 2020). FHE allows the users to perform cryptographic process 'n' times. FHE comprises of four main processes.

Figure 1: Proposed Key generation process using GA


Random set of Initial Population
Enter the desired key length in bits (16,32,64.128)
Apply Fitness evaluation (Gap test & Frequency test)
Apply Uniform Crossver operator
Apply BitFlip Mutation operator
Generate set of candidate keys
Apply Autocorrelation test, Select the keys with nearer to 0 relativity
Generate the Final key for Encryption process

1. Key Generation Process: generating key for symmetric/ asymmetric HE methods.

2. Encryption Process: Generates ciphertext C from the plaintext P, C=E(P) with the encryption key K

3. Decryption Process: Decrypts the ciphertext to generate the plaintext D(C)=P with the same key.

4. Evaluation Process: Performs the functional operations like additive or multiplicative over the ciphertexts C1, C2 and evaluate the output by Fn(C1, C2)=E(Fn(P1,P2), it means D(Fn(C1,C2)=Fn(P1,P2) without even knowing the plaintext P.

Proposed Algorithm

2.1 Symmetric Key Generation Using Genetic Algorithm: The first phase of the proposed work is to generate non-repetitive, highly unpredictable random cryptographic key using GA for the symmetric Homomorphic Encryption system Random number generator is utilized to populate the initial population of GA. The keys are encoded to decimal numbers. The crossover and mutation operators are applied to the current population to produce the chromosomes for the next generation. In each sequence of key generated the relationship amongst the chromosomes are evaluated using the statistical tests in sequence. The evaluation fitness function for the key selection among the candidate keys will be the combination of gap, frequency and auto correlation tests (Mahender et al, 2017 and Ranjeeth et al 2020). Depending on number of iterations these processes are be repeated N times. The sequence of chromosomes with the correlation value close to 0 is selected as the cryptographic key for FHE. The proposed GA based key generation method is shown as a flowchart in Figure:1.

Generating Initial Population: The population size is taken as 250 and the number of generations is taken as 1000. The first parent chromosomes are generated using PRNG which is followed by the iterative process of applying crossover and mutation operators to generate the next generation chromosomes.

Crossover operator: The parents generated in the new population are treated with Single point crossover. Random crossover point is chosen and performs swapping among the chromosomes of parent generations to produce set of new children or offspring. These children are dissimilar from the previous generations. The statistical tests, Gap and Frequency tests are applied to the set of new offspring. The population with minimum frequency occurrence and maximum gap between occurrences is selected as candidate population. 0.2% crossover rate is taken for the above-mentioned processes.

Mutation Operator: Mutation is the process of flipping or altering one or more gene in the selected set of chromosomes. With a mutation rate of 0.3%, Bit Flip mutation is applied to the population which inverse the bits. A bit '1' is inversed to '0' and viceversa. At the end of the mutation process the most fit chromosomes replaces the less fits.

Fitness Evaluation Function: This is the objective function to evaluate the fitness of the generated chromosomes. This function determines how close the current population is to the goal state in the problem space. The individual chromosomes with the highest fitness value are selected from the initial population to be treated with the other GA operators. The individual chromosome generated is tested for the number of occurrences by using Frequency test. And the consequences of the interval between the repeated occurrences is tested using Gap test. The chromosomes with minimum occurrences and maximum gap are selected as the candidate key and stored in a repository. In order to generate a strong, non-repetitive and unpredictable keys for the cryptographic operations in FHE, the stored key are again evaluated with Auto correlation test which examines the relationship between any two bits in the keys. The values lie between +1.00 and -1.00 which is positive correlation and negative correlation. This linear property evaluates the randomness of the genes in the population. The chromosomes with near zero value indicate high randomness and those chromosomes are chosen and generated as the key for the encryption operation for FHE.

Fhe Scheme With Symmetric Key Encryption: The key which is generated using GA is of16, 32, 64 and 128 bits length depending on the symmetric encryption scheme used for FHE. The plaintext and the symmetric key are mapped as matrix in order to achieve the homomorphism. The proposed FHE scheme is based over Integers which search for the Approximate Greatest Common Divisor method (Coron et al., 2011, Wainakh et al. 2018) which has efficient homomorphic property. The proposed Symmetric FHE is as follows:

Step 1: The key generated from GA is taken as input for the FHE process.

Step 2: Symmetric key selection: Choose the key X that is an odd integer from certain intervals, such that X∈ (2N, 2N-1).

Step 3: Encryption Process: Let M be the bit from the plaintext P, to encrypt bit M, take M ∈ {0,1}, the ciphertext C is set as C= Encrypt(M) = M+ 2R +XY, where Y and R are random numbers. R is chosen with the condition R < X/2.

Step 4: Decryption Process: The ciphertext is decrypted as Decrypt(C) = (C(Mod X))(Mod 2)=M.

Step5: Evaluation Process: Fully Homomorphism can be achieved by applying Addition and Multiplication operations over the given integers. Let C is the Ciphertext, Ci are the ciphers which are partitioned.

Step 5.1: Apply addition,

$$\Rightarrow \quad M_1+2R_1 + XY_1 + M_2 +2R_2 + XY2$$
$$\Rightarrow \quad (M_1 +M_2) + 2(R_1+R_2) +X(Y_1+Y_2)$$
$$\Rightarrow \quad \text{Encrypt}(M_1+M_2)$$

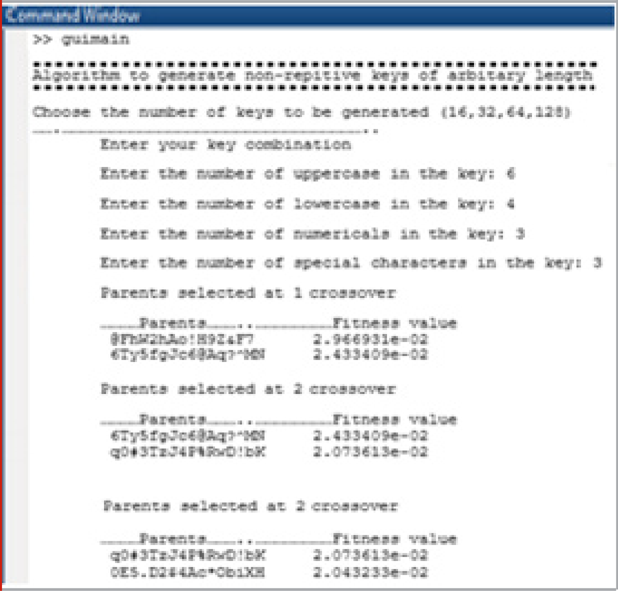
Step 5.2: The ciphertext C can be decrypted if the condition 2R1R2 +M1R2 +M2R1< X/2

Step 6: As long as step 5.2 is true, then the plain text is retrieved from C.

The symmetric FHE needs varying key length as input for different types of symmetric cryptosystem. For example, AES-128 symmetric algorithm needs the key length of 16 bits to match with the input block size.

3. Experimental Result Analysis: The implementation of the proposed algorithm explained in section 2 has been done in MATLAB to analyze the result. For the experimental purpose the proposed system is tested for text files and .wav audio files. The key generation process plays a vital role in this proposed work. The strength of any cryptosystem lies in the strength of the key used for encryption. The result proves that the GA process can generate a fittest and strong key for the symmetric encryption process. Figure 2 shows the snapshot of the process that happens during key generation. Figure 3 shows the possible keys generated with fitness values. Figure 4 shows the fittest secure key that is chosen among the possible candidate keys after the application of statistical test.

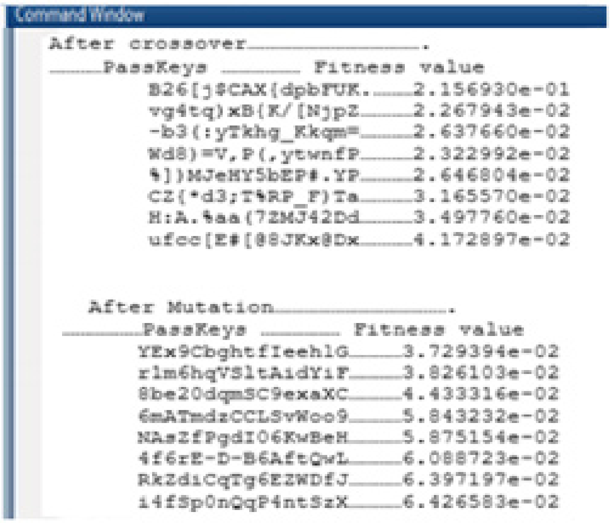
Figure 2: Snapshot of the process that happens during 16bit keygeneration.



```
>> guimain

**************************************************************
Algorithm to generate non-repitive keys of arbitary length
**************************************************************

Choose the number of keys to be generated (16,32,64,128)
---.-----------------------------------------------..
          Enter your key combination

          Enter the number of uppercase in the key: 6

          Enter the number of lowercase in the key: 4

          Enter the number of numericals in the key: 3

          Enter the number of special characters in the key: 3

          Parents selected at 1 crossover

          ______Parents______..____________Fitness value
           @FhW2hAo!H9Z&F7        2.966931e-02
           6Ty5fgJc6@Aq?^MN       2.433409e-02

          Parents selected at 2 crossover

          ______Parents______..____________Fitness value
           6Ty5fgJc6@Aq?^MN       2.433409e-02
           q0#3TzJ4P%RwD!bK       2.073613e-02


          Parents selected at 2 crossover

          ______Parents______..____________Fitness value
           q0#3TzJ4P%RwD!bK       2.073613e-02
           0E5.D2#4Ac*Ob1XH       2.043233e-02
```

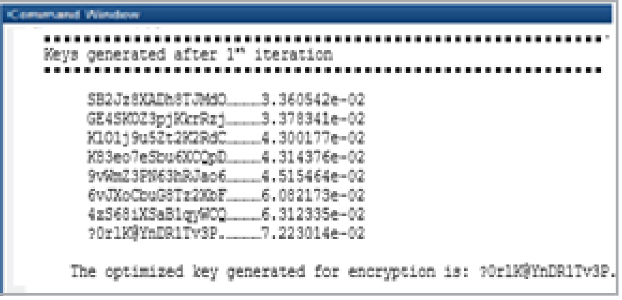
The experiment has been conducted with the GA control parameters with population size 16, 32,64 and 128 in order to generate the encryption keys which can be chosen with respect to the symmetric encryption algorithm and the generations as 250, 500, 750 and 1000. The throughput time or the execution time is given in Table 1. The variation in key length and variation in the number of iterations has an impact on the execution time for the key generation as shown. But due to the complete randomness nature of the key the slight increasein running time can be tolerable.

Figure 3: Possible keys generated with fitness values



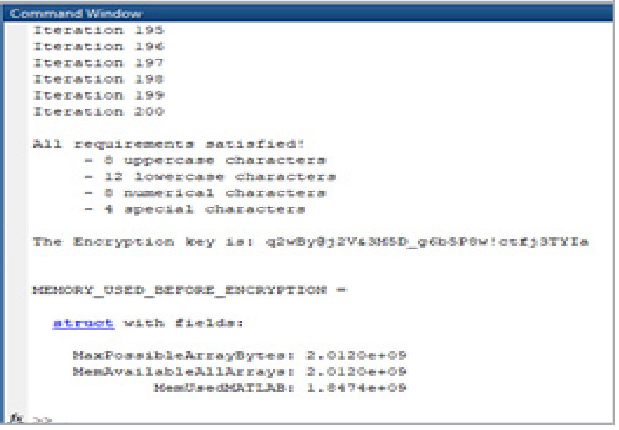
```
   After crossover_________________________.
   ______PassKeys _________ Fitness value
             B26[j$CAX{dpbFUK.______2.156930e-01
             vg4tq)xB(K/[NjpZ_______2.267943e-02
             -b3(:yTkhg_Kkqm=_______2.637660e-02
             Wd8)=V,P(,ytwnfP_______2.322992e-02
             %])MJeHY5bEP#.YP_______2.646804e-02
             CZ{*d3;T%RP_F)Ta_______3.165570e-02
             H:A.%aa(7ZMJ42Dd_______3.497760e-02
             ufcc[E#[@8JKx@Dx_______4.172897e-02


      After Mutation_________________________.
    __________PassKeys _________ Fitness value
            YEx9CbghtfIeehlG_______3.729394e-02
            rlm6hqVSltAidYiF_______3.826103e-02
            8be20dqmSC9exaXC_______4.433316e-02
            6mATmdzCCLSvWoo9_______5.843232e-02
            NAsZfPgdI06KwBeH_______5.875154e-02
            4f6rE-D-B6AftQwL_______6.088723e-02
            RkZdiCqTg6EZWDfJ_______6.397197e-02
            i4fSp0nQqP4ntSzX_______6.426583e-02
```

Figure 4: The fittest key chosen for the symmetric FHE



```
   **************************************************************
   Keys generated after 1st iteration
   **************************************************************

       SB2Jz8XADh8TJMdO______3.360542e-02
       GE4SKDZ3pjKkrRzj______3.378341e-02
       K1O1j9u52t2K2RdC______4.300177e-02
       K83eo7eSbu6XCQpD______4.314376e-02
       9vWm23PM63hRJao6______4.515464e-02
       6vJXoCbuG8Tz2XbF______6.082173e-02
       4z568iXSaB1qyWCQ______6.312335e-02
       ?0rlK@YnDR1Tv3P.______7.223014e-02

     The optimized key generated for encryption is: ?0rlK@YnDR1Tv3P.
```

Figure 5: A sample 32 bits key generated for encryption



```
Iteration 195
Iteration 196
Iteration 197
Iteration 198
Iteration 199
Iteration 200

All requirements satisfied!
     - 8 uppercase characters
     - 12 lowercase characters
     - 8 numerical characters
     - 4 special characters

The Encryption key is: q2wBy8j2V&3H5D_g6b5P8w!ctfj3TYIa


MEMORY_USED_BEFORE_ENCRYPTION =

  struct with fields:

    MaxPossibleArrayBytes: 2.0120e+09
    MemAvailableAllArrays: 2.0120e+09
            MemUsedMATLAB: 1.8474e+09

>>
```

Table 1. Time taken to generate keys of various size

KEY SIZE (POPULATION SIZE)	ITERATIONS (NUMBER OF GENERATIONS RUN)			
	250	500	750	1000
16	0.1670	0.2140	0.3331	0.5709
32	0.1967	0.2993	0.5675	1.0832
64	0.2763	0.4930	0.8356	1.4760
128	0.4375	0.7010	1.4112	2.9370

Table 2. Cryptographic time analysis for input text files

TEXT FILE SIZE IN KB (approx.)	ENCRYTPTION TIME (MS)	DECRYPTION TIME (MS)	EVALUATION TIME (MS)
10	1.3231	0.8463	4.0113
20	3.6958	2.9910	9.3751
40	8.0339	6.4721	20.9135

Table 3. Cryptographic time analysis for input Audio files

AUDIO FILE SIZE IN KB (approx.)	ENCRYTPTION TIME (MS)	DECRYPTION TIME (MS)	EVALUATION TIME (MS)
39	11.0329	8.1675	14.6300
283	36.1938	27.0331	39.1492

The proposed GA can generate keys of size 16,32,64 and 128 bits. Key generated for 32 bits encryption scheme for the 200 iteration is shown in Figure 5. Table 1 gives the time taken by the proposed system to generate different key length with respect to the population sizes. For the experimental simulation purpose, the time taken for the cryptographic process and the evaluation time has been calculated for the various file sizes of 10KB, 20KB and 40KB taking text files as the input. Random audio files of 39KB and 283 KB were taken as input. The results we shown in Table 2 and Table 3 respectively.

The evaluation time is calculated by the FHE with the generated symmetric key as the encryption key.

CONCLUSION AND FUTURE WORK

The growing usage of the cloud computing application increases the importance of a more robust encryption scheme like Fully Homomorphic Encryption. The efficiency of any cryptographic system lies in generation of strong and robust keys which are very difficult to tamper. Hence the emphasis is on generating the fittest keys for the encryption process. So, this paper justifies that GA generates the non-repetitive, unpredictable complex keys for symmetric cryptosystem that enhances the security of symmetric FHEs. This work can be extended to be applicable for public key cryptosystem also. Integer based FHE scheme is used in this paper. Other FHE schemes can be implemented for evaluation process and an optimized FHE method can be identified with respect to cryptographic and evaluation time. Though the key generated by GA is an optimized key, comparison can be done with other evolutionary approaches enhancing the FHE to make the system more precise and stronger.

REFERENCES

Bensitel, Y. and Romadi, R., 2016, May. Secure data storage in the cloud with homomorphic encryption. In 2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech) (pp. 1-6). IEEE.

Coron, J.S., Mandal, A., Naccache, D. and Tibouchi, M., 2011, August. Fully homomorphic encryption over the integers with shorter public keys. In Annual Cryptology Conference (pp. 487-504). Springer, Berlin, Heidelberg.

Gentry, C., 2009. A fully homomorphic encryption scheme (Vol. 20, No. 9, pp. 1-209). Stanford: Stanford university.

Goldberg, D.E., 1989. Genetic algorithms in search, optimization, and machine learning. Boston, MA: Addison-Wesley.

Kalaiselvi, K. and Kumar, A., 2017. Effect of variations in the population size and Generations of Genetic Algorithms in Cryptography-An Empirical Study. Indian Journal of science and technology, 10(19), pp.1-6.

Mahender, K., Ramesh, K.S. and Kumar, T.A., 2017. An efficient ofdm system with reduced papr for combating multipath fading. Journal of Advanced Research in Dynamical and Control Systems, 9(Special issue 14), pp.1939-1948.

Ranjeeth, S. and Latchoumi, T.P., 2020. Predicting Kids Malnutrition Using Multilayer Perceptron with Stochastic Gradient Descent. Rev. d'Intelligence Artif., 34(5), pp.631-636.

Sindhuja, K. and Devi, S.P., 2014. A symmetric key encryption technique using genetic algorithm. International journal of computer science and information technologies, 5(1), pp.414-416.

Van Dijk, M., Gentry, C., Halevi, S. and Vaikuntanathan, V., 2010, May. Fully homomorphic encryption over the integers. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 24-43). Springer, Berlin, Heidelberg.

Wainakh, A., 2018. Homomorphic encryption for data security in cloud computing (Doctoral dissertation, MIDDLE EAST TECHNICAL UNIVERSITY).

William, S., 2011. Cryptography and network security: principles and practice.

Yookesh, T.L., Boobalan, E.D. and Latchoumi, T.P., 2020, March. Variational Iteration Method to Deal with Time Delay Differential Equations under Uncertainty Conditions. In 2020 International Conference on Emerging Smart Computing and Informatics (ESCI) (pp. 252-256). IEEE.