ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 10(19), DOI: 10.17485/ijst/2017/v10i19/110803, May 2017* ISSN (Online) : 0974-5645

Effect of Variations in the Population Size and Generations of Genetic Algorithms in Cryptography - An Empirical Study

K. Kalaiselvi[1*] and Anand Kumar[2]

[1]Department of Computer Science, Kristu Jayanti College, Bangalore – 560077, Karnataka, India; kalaiselvi76@yahoo.com
[2]Department of Computer Science and Engineering, M. S. Engineering College, Bangalore – 562110, Karnataka, India; kumaranandkumar@gmail.com

Abstract

Objectives: The implementation of Genetic algorithm in the symmetric block cipher Advanced Encryption Standard -128 (AES-128) algorithms to enhance the performance of cryptographic operations. **Methods:** Genetic algorithm is used for generating the best fit non-repetitive cipher key and for key distribution to design a dynamic Substitution box in AES-128. **Findings:** The study reveals that the efficiency of the cryptographic algorithm treated with Genetic algorithm is dependent on the variations in the number of generations and initial population size. The result shows that an optimum population size has less encryption and decryption time. Among the sample population size taken for the experiment, almost the average population size has minimum encryption and decryption time. Results from iteration variations shows that the average number of iterations has less encryption and decryption time. **Improvements:** The hybrid combination of Genetic algorithm and AES-128 can be further modified for images and audio messages also.

Keywords: Genetic Algorithm, Iterations, Non-Repetitive Cipher Key, Population Size, Substitution Box

1. Introduction

Cryptography offers security and protection for the data when it is communicate through the network. In order to achieve high security, the data is encrypted at the sender side and again decrypted in the receiver side using either a same key or a pair of different keys. Encryption and decryption are performed using cryptographic algorithms which can be either symmetric or asymmetric according to the types of keys used. Symmetric algorithm uses same key for both encryption and decryption. Among many algorithms developed Advanced Encryption Standard (AES) is proven to be stronger in terms of cryptanalytic attacks[1]. The research work has been undertaken to implement the concept of Genetic Algorithm (GA) in S-box of AES algorithm[2]. The experimental result shows that the control parameters of GA play an important role in cryptographic operations. The focus of this paper is to provide the experimental detail that proves the variations in the parameters like population size and number of generation has an impact on the encryption and decryption time of the enhanced AES algorithm using GA.

2. Structure of AES

The AES algorithm belongs to the symmetric cipher in which the cryptographic process uses a single key for both encryption and decryption. The key length is equal to the input block size. AES-128 has 10 rounds of each encryption and decryption process.

There are four stages in each round. They are Substitution, Transposition, Mixing input plaintext and converting it to cipher text. Conversion of cipher text into plaintext follows the reverse rounds process using the same key. The strength of AES depends on the confusion (substitution) and diffusion (permutation). The inverse round is performed for decrypting the cipher text into plain text. The working structure of AES is shown in Figure 1.

* *Author for correspondence*

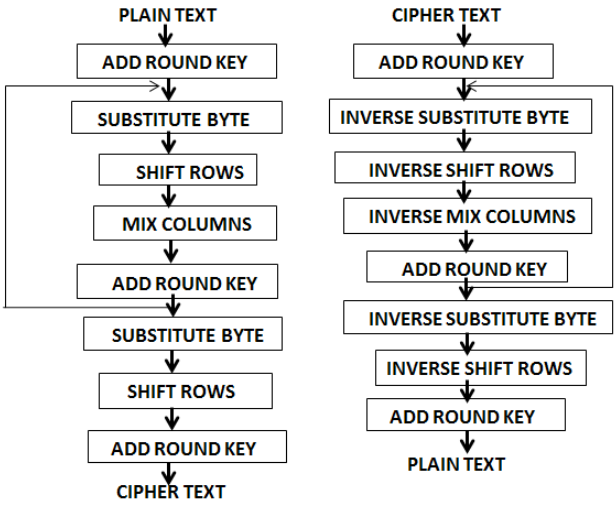

Figure 1. Working structure of AES.

3. Genetic Algorithm

Genetic algorithms are based on natural selection and natural genetics, which combines the survival of the fittest mechanism with a randomized information exchange to form an optimized search algorithm[3]. Genetic algorithms provides an optimized solution available in the search space. A conventional GA is composed of populations, fitness function and operators which controls the process. The process of GA is shown in Figure 2.

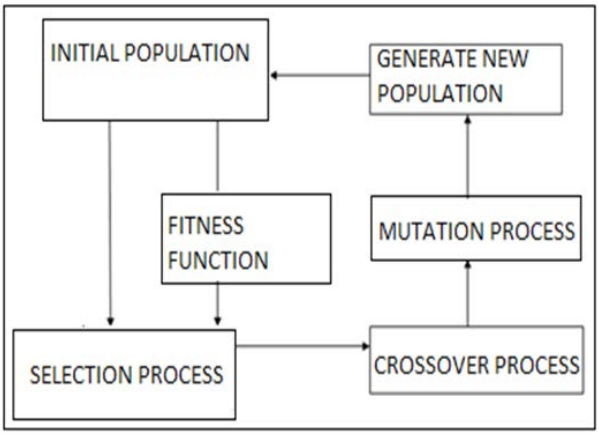

Figure 2. Genetic algorithm process.

The pseudo code for a simple genetic algorithm can be described as follows.

Begin

Generate the initial population randomly.

Calculate the fitness value.

Repeat

Selection: a pair of parents based on fitness value.

Crossover: Create two offspring.

Mutation: Apply to each child.

Evaluate: Mutate percentage.

New population: All the offspring are new.

Until (Terminate condition = TRUE).

End

The initial population is generated randomly and the fitness criteria are applied to it to select the first generation chromosomes. The GA operators are performed to generate new set of chromosomes; the generated result is added to the next generation. The process continues till the optimized result is reached.

3.1 Basic Terms in Genetic Algortihm

In GA, the term *chromosome* typically refers to a candidate solution to a problem. The high quality candidate solution from different regions of the search space is referred as *parent*. The combination of chromosomes termed as *population* is randomly generated at the initial stage of the algorithm. The chromosomes are represented either as binary (0 or 1) or hex number depending on the type of population. They are iterative in nature and transform the population into a new generation. GA uses a fitness function which assigns a score to each chromosome of the current population. The individuals are selected by applying different combination of genetic operators.

3.2 Genetic Operators

A simple genetic algorithm uses three basic operators: namely selection, crossover and mutation.

Selection:

The operator selects the chromosomes in the population for reproduction according to the fitness function.

Crossover:

The operator chooses the position of bits and exchanges the chromosomes between the chosen positions.

Mutation:

The operator randomly flips the bits in a chromosome, (i.e.,) from 0 to 1 or 1 to 0.

3.3 Parameters of Genetic Algorithm

There are three major parameters that determine the

efficiency of GA: 1. Encoding method. 2. Type of operator used and 3. Control parameters. These control parameters refers to the population size and the number of generations which are randomly generated to apply the operators[4,5]. The proposed algorithm uses binary coding which is the general encoding method. For efficiency uniform crossover, Roulette Wheel Selection and Inverse Mutation are used to develop the algorithm. The proposed work concentrates on analyzing the variations in the control parameter with respect to the efficiency (i.e.,) encryption time, decryption time and throughput time of AES-GA algorithm.

4. Proposed Algorithm

The proposed algorithm uses the Genetic Algorithm process in AES to strengthen the key generated by AES-128. The strength of AES lies in combination and permutation of the key generated. This is achieved by applying GA in the process of diffusion and confusion[6,7]. The randomly generated initial population and the GA operators produce non-repetitive combination of encryption keys which makes the cryptanalytic attack difficult. This research paper proposes an approach to generate unique keys by using the pseudo random number generator. The basic processes of GA are used to complicate the key generated by the 128-bit key AES[8]. The GA process is utilized to strengthen the key generated by AES-128 which is expected to increase the data parameters[9] with respect to the cryptographic time and the throughput time of the AES-GA. The proposed algorithm is shown in simple steps.

The proposed AES-GA algorithm:
Step 1: Input plain text
Step 2: Generate the cipher keys using Pseudo Random number generators
Step 3: Apply Genetic algorithm operators
Step 4: Evaluate the keys using fitness function
Step 5: Choose the best fit key from the generated population
Step 6: Design the Substitution box for AES-128
Step 7: Encryption process –Cipher text
Step 8: Decryption process- Plain text
Step 9: Stop

5. Varying Population and Generations

The process of GA starts with a randomly generated population keys which are known as chromosomes. The length of the key determines the number of genes (bits) in the population. This proposed work uses 128-bit key size. The generated initial population will be treated with the genetic operators which increases the total number of chromosomes. The individuals are selected according to the fitness value. The selection operator is impacted due to the variation in the population size. Basically, the GA starts with the fixed population size and fixed number of iterations. The proposed work uses Roulette wheel selection, inversion mutation and uniform crossover with constant probability for the operators. The fitness probability is maintained as constant value[10]. A GA with varying population size and varying iterations[11] requires very less encryption time, decryption time and throughput time in comparison with the constant control parameters GA.

6. Experimental Result and Analysis

The technique which is explained in section III has been implemented using Matlab and the observations are compared and analyzed. The GA parameters are initialized as:

Fitness function: Sine x
Probability of crossover (P_c): 0.8
Probability of mutation (P_m): 0.3
Number of iterations: 100
Population size: varying

6.1 Variation in the Population Size

The population size is taken to be varying. The sample size taken for this research paper is 10, 50, 75, 100, 200, 300, 500, 750 and 1000. Rest all the parameters are considered as constant values. Table 1 shows the changes in memory usage, encryption and decryption time due to the variation in the population size.

Table 1. Values for varying population size

Population size	Memory usage-Encryption (Bytes)	Memory usage -Decryption (Bytes)	Encryption Time (seconds)	Decryption Time (seconds)
10	7.41028	7.34441	3.07	0.393
50	7.31779	7.29432	3.12	0.394
75	7.43404	7.40319	3.04	0.391
100	7.28551	7.26446	3.08	0.396
200	7.16567	7.18655	3.04	0.391
300	7.20212	7.19585	3.10	0.391
500	7.47409	7.46922	3.04	0.392
750	7.43567	7.43342	3.12	0.398
1000	7.44759	7.44575	3.11	0.392

By varying the population sizes, other evaluating functions like crossover and mutation are kept as constant. The result shows that there is an optimum population size, beyond or below which the performance of GA in cryptographic operation degrades. For a large population size the time taken for encryption and decryption are more compare to the small population size. In the sample population set taken, for the population size of 200, the encryption time and decryption time is comparatively less than the minimum population size 10 and the maximum population size 1000. The performance has reached the peak for the100 generations and 200 population size. An optimal solution will be reached with varying P_c and P_m which will be performed as the continuation in this research. The result is shown as a graph, plotted with the population size and encryption time and decryption time. Figure 3 shows the result graph of the encryption time with varying population. Figure 4 shows the result graph of decryption time with varying population.

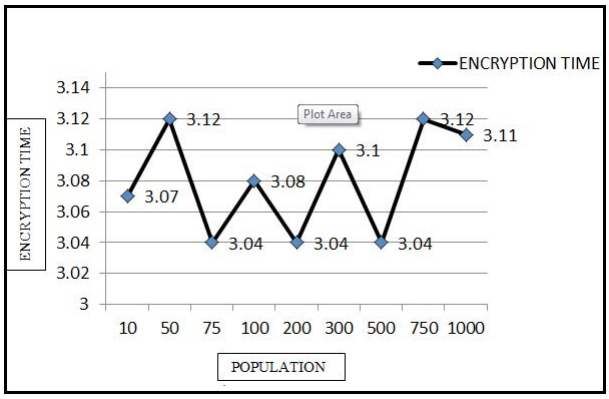

Figure 3. Encryption time graph with varying population.

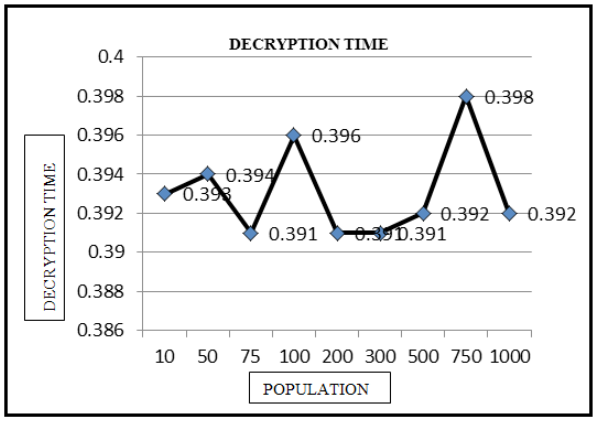

Figure 4. Decryption time graph with varying population.

6.2 Variation in the Number of Iterations

The number of iterations (generations) are varying. The sample iterations are taken as 10, 30, 100, 150, 200, 250, 300 and 350. The GA parameters are initialized as:

Fitness function: Sine x
Probability of crossover (P_c): 0.8
Probability of mutation (P_m): 0.3
Population size: 250
Number of iterations: varying

Table 2 shows the performance of GA with respect to memory usage, encryption and decryption time due to the variation in the number of iterations.

The result shows that the cryptographic time is less when the number of iterations is neither too large nor too small. With respect to the sample iterations taken,

Table 2. Values for varying number of iterations

Generations (Iterations)	Memory usage-Encryption (Bytes)	Memory usage -Decryption (Bytes)	Encryption Time (seconds)	Decryption Time (seconds)
10	7.48298	7.48188	3.04	0.396
30	7.36465	7.37747	3.06	0.391
100	7.43023	7.43023	3.05	0.396
150	7.43096	7.43088	3.05	0.391
200	7.44202	7.43608	3.03	0.391
250	7.51968	7.50440	3.09	0.39
300	7.51624	7.51186	3.10	0.39
350	7.36465	7.37747	3.06	0.39

iteration size of 200 with the population size of 250 has comparatively less encryption and decryption time. The optimal result can be achieved by varying Pc and Pm with suitable crossover and mutation operators.

Figure 5 shows the result graph of the encryption time with varying iterations. Figure 6 shows the result graph of decryption time with varying iterations.

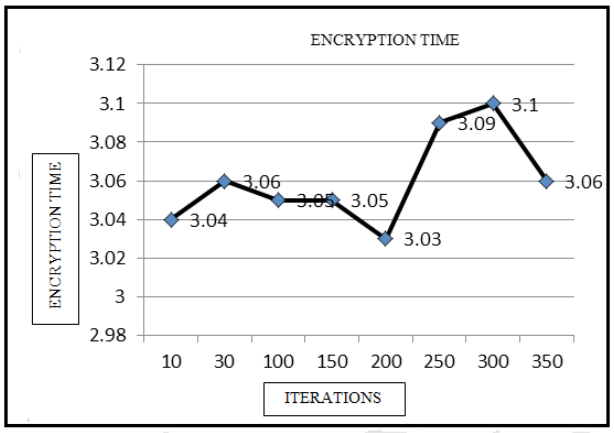

Figure 5. Encryption time graph with varying iterations.

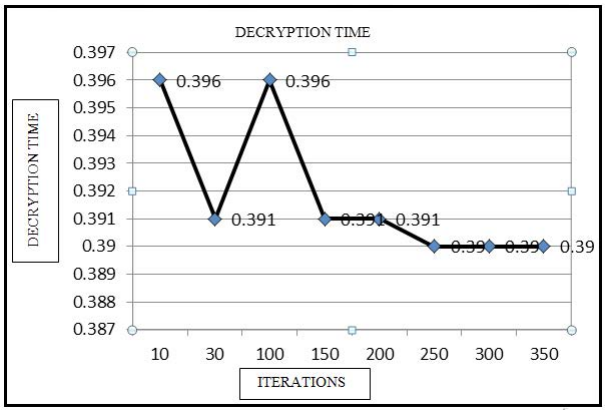

Figure 6. Decryption time graph with varying iterations.

7. Conclusion

The effect of changes in GA parameters with respect to the encryption time and decryption time of AES -128 has been studied in this research paper. Keeping the evaluation function of crossover and mutation, cryptographic key size as constant, it is observed that the cryptographic time is varying depending only on the variation in population size and number of iterations. The aim of this paper is not to project the optimal population size and iteration size for AES-128 cryptosystem, but to reach the general conclusion. The following conclusions can be derived from the study:

- The result shows that neither a very small population nor a comparatively large population has less encryption and decryption time. Among the sample population size taken for the experiment, almost the average population size has minimum encryption and decryption time.
- Results from iteration variations shows that the average number of iterations has less encryption and decryption time.
- To achieve good performance and improved efficiency of AES-128, crossover rate should be high with low mutation rate combined with the optimal population size and iterations.

8. Future Work

The research work has been undertaken to enhance the performance of AES-128 bit cryptosystem using Genetic algorithm. GA will be used for key generation and key distribution which efficiently uses the varying population and varying iterations during the process of encryption and decryption. Further, the research result will give the

optimal crossover and mutation operator combination which will increase the efficiency of AES.

9. References

1. Stallings W. Cryptography and network security. Principles and Practice. 3rd ed. Prentice Hall; 2007.
2. Kalaiselvi K, Kumar A. Enhanced AES Cryptosystem by using Genetic Algorithm and Neural Network in S-box; 2016. p. 1–3.
3. Goldberg DE. Genetic algorithms in search, optimization and machine learning. New York: Addison Wisley; 1989. p. 1–6.
4. Kapoor V, Dey S, Khurana AP. An empirical study of the role of control parameters of genetic algorithms in function optimization problems. International Journal of Computer Applications. 2011Oct; 31(6):20–6.
5. Goldberg DE, Deb K. A comparative analysis of selection schemes used in genetic algorithmIn: Rawlins, Gregory JE, Editor. Foundations of Genetic Algorithms. Morgan Kaufmann Publishers Inc; 1991. p. 69–93.
6. Tragha A, Omary FA, Mouloudi. ICIGA. Improved Cryptography Inspired By Genetic Algorithms. 2006, 3(2), pp. 1-3.
7. Batina L, Jakobovic D, Mentens N, Picek S, Piedra AD, Sisejkovic D. S-box pipelining using genetic algorithms for high-throughput AES implementations. 2014; 8885:322–37.
8. Sindhuja K, Devi PS. A symmetric key encryption technique using genetic algorithm. International Journal of Computer Science and Information Technologies. 2014; 5(1):414–6.
9. Arrag S, Hamdoun A, Tragha A, Khamlich SE. Replace AES key expansion algorithm by modified genetic algorithm. Applied Mathematical Sciences. 2013; 7(144):7161–71. Crossref
10. Eiben AE, Hinterding R, Michalewicz Z. Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation.1999; 3(2):124–41. Crossref
11. Boyabalti O, Sabuncuoglu I. Parameter selection in genetic algorithms. Systemics, Cybernetics and Informatics. 2007; 2(4):78–83.