International Journal of Security and Its Applications
Vol.9, No.9 (2015), pp.295-304
http://dx.doi.org/10.14257/ijsia.2015.9.9.26

A Genetic Algorithm Approach for Breaking of Simplified Data Encryption Standard

Farah Al Adwan[1], Mohammad Al Shraideh[2] and Mohammed Rasol" Saleem Al Saidat[3]

[1, 2]*Department of Computer Science, University of Jordan*

[3]*Jordan Telecom Group (Orange)*

farah.aladwan14@yahoo.com, mshridah@ju.edu.jo, mohammedrasol@gmail.com

Abstract

A genetic algorithm (GA) is a search algorithm for solving optimization problems due to it is robustness; it offers benefits over optimization techniques in searching n-dimensional surface. In today's information age, information transfer has increased exponentially. Hence, security, confidentiality and authentication have become important factors in multimedia communications. Encryption is an effective technique that is preserving the confidentiality of data in Internet applications. Cryptanalysis is a technique of encoding and decoding ciphertext in such way it cannot be interpreted by anyone expects sender and receiver. In this paper, GA with an improved crossover operator was used for the cryptanalysis of Simplified data encryption standard problem (S-DES). Results have shown that GA performance is better than brute force search technique in breaking S-DES key.

Keywords: *Genetic algorithm, Cryptanalysis, S-DES, Key search space, n-Grams*

1. Introduction

Cryptology is divided into two parts: Cryptography and Cryptanalysis. Cryptography concerns itself with the design of Cryptosystems while Cryptanalysis is the study of breaking ciphers; this involves finding the key that is used for encrypting the message or recovering a plaintext from ciphertext without a key. In order to prevent attacks, Cryptographic systems have a large key space which means remaining the systems secure from attacks because the size of the key space. For this reason, many meta-heuristics such as GA [1-2] have been used in determining efficient solutions of complex problems like cryptanalysis of S-DES.

The attacker in the brute force attack tries every possible key on the part of ciphertext until determine the correct key that used to encrypt the ciphertext. Therefore, it is difficult to break the cipher using brute force attack since the difficulty of breaking the cipher is directly proportional to the number of keys. But, GA is an efficient optimization technique can be used to solve breaking key problem and takes less time as compared to brute force attack. On other hand, random search algorithm belongs to numerical optimization family; it can be applied on block-box optimization problems [3]. Thus, it considers good choice to solve breaking problems beside GA. The system work of random search is simple: a random solution is selected at each iteration from the search space. If the selected solution is better than the current solution, this solution replaced with the current solution. Finally, the best solution found by the random search algorithm is returned.

The rest of paper is organized as follows: Section 2 describes the related works done in this field, Section 3 presents an overview of S-DES and Section 4 gives the overview

ISSN: 1738-9976 IJSIA

of Genetic Algorithm. Results and Experimental setup are discussed in Section 5. Conclusions of our study are presented in Section 6.

2. Related Works

There are several solutions that have been published about the using of GA to cryptanalyze ciphertext. Authors in [4] introduce a combination of GA and particle swarm optimization methods with the intent if breaking DES. The proposed method computes a feasible set of optimum keys with a given plaintext without searching in the entire key space. Side-channel analysis (SCA) is one of the latest cryptanalysis attacks that is reported by [5] authors, the proposed technique exploits information leaking from electronic devices by applying the machine learning techniques to cryptanalysis. Recently, [6] proposed a new GA based on secret key image encryption method by exploitation the powerful feature of the mutation and crossover operations, the results show that the presented method is satisfied the goals that are need in any encryption method. In [7] Presented a new approach using a niche genetic algorithm that can effectively get an optimal solution by reduce the complexity of the problem analysis. In [8] a novel approach called Genetic Swarm Optimization (GSO) has proved by combining the effectiveness of GA and Particle Swarm Optimization (PSO) to attack S-DES. The results showed that GSO reduces the key search space by the factor of 5.6. In [9] authors proposed a Genetic algorithm approach by using Ring crossover and other operators for the cryptanalysis of S-DES. The results showed that the genetic Algorithm is far better than Brute Force search algorithm for cryptanalysis of S-DES. Authors In [10] proved the success of Genetic Algorithm over brute force in the cryptanalysis of the cipher text by mentioned the causes for the success of Genetic Algorithm and its basic operations.

3. S-DES

S-DES is a simplified version of the Data Encryption Standard (DES). This algorithm is not cryptographically secure, but it is for educational purposes only. It was originated by Edward Schaefer, the professor at Santa Clara University [11].

The S-DES encryption algorithm takes an 8-bit block of plaintext (example: 10111101) and 10-bit key as input and produces an 8-bit block of cipher text as output, while the S-DES decryption algorithm takes an 8-bit block of cipher text and the same 10-bit key used to produce that cipher text as input and produces the original 8-bit block of plaintext [12, 13].

3.1. S-DES Key Generation

S-DES depends on the use of a 10-bit key shared between sender and receiver. From this key, two 8-bit sub keys are produced through different binary operations: circular left shift and permutations, for use in particular stages of the encryption and decryption algorithm as shown in Figure 1 [14].

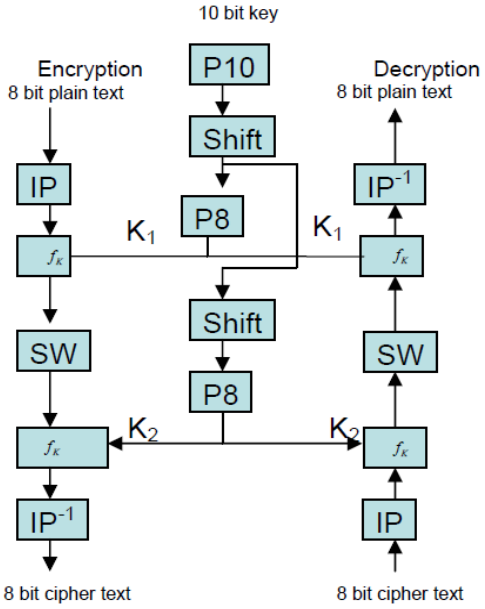

Figure 1. Simplified DES Algorithm

Encryption involves the sequential application of five functions: an initial permutation (IP), a complex function labeledf_k, a simple permutation function that switches (SW) the two halves of the data, the function f_k again and finally a permutation function that is the inverse of the initial permutation(IP^{-1}). These different steps are now briefly detailed:

3.2. Initial and Final Permutation

The algorithm takes as input an 8-bit block of plaintext, which we first permute using the IP function IP= [2 6 3 1 4 8 5 7]. This retains all 8-bits of the plaintext but mixes them up. At the end of the algorithm, the inverse permutation is used by applying, IP^{-1} = [4 1 3 5 7 2 8 6].

3.3. The Functionf_k

The most complex component of S-DES is the function f_k, which consists of a combination of permutation and substitution functions. The functions can be expressed as follows:

Let L and R be the leftmost 4 bits and rightmost 4 bits of the 8-bit input tof_k, and let F be a mapping from 4-bit strings to 4-bit strings. Then we let:

$$f_k(\text{L, R}) = (\text{L XOR F(R, SK)}, \text{R})$$

Where F is a mapping from a 4-bit strings to a 4-bit strings, L is the leftmost 4 bits, R the Right most 4 bits, SK is a sub key and XOR is the bit-by-bit exclusive OR function.

The mapping F is the trickiest part of the scheme as it involves nonlinear functions, as described below:

1. The first step is known as an expansion/permutation operation E/P= [4 1 2 3 2 3 4 1] to input 4-bits ($n_1n_2n_3n_4$). For what follows, it is clearer to depict the result in this fashion:

$$\left|\begin{matrix} n_4 & n_1 \\ n_2 & n_3 \end{matrix}\right| \left|\begin{matrix} n_2 & n_3 \\ n_4 & n_1 \end{matrix}\right|$$

2. The 8-bit sub key K1= $(k_{11}k_{12}k_{13}k_{14}k_{15}k_{16}k_{17}k_{18})$ is added to this value using exclusive-OR:

$$\left|\begin{matrix} n_4 \oplus k_{11} & n_1 \oplus k_{12} \\ n_2 \oplus k_{15} & n_3 \oplus k_{16} \end{matrix}\right| \left|\begin{matrix} n_2 \oplus k_{13} & n_3 \oplus k_{14} \\ n_4 \oplus k_{17} & n_1 \oplus k_{18} \end{matrix}\right|$$

3. The first 4 bits (first row of the preceding matrix) are fed into the S-box S0 to produce a 2-bit output, and the remaining 4 bits (second row) are fed into S1 to produce another 2-bit output. These two boxes are defined as follows:

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

The S-boxes operate as follows. The first and fourth input bits are treated as a 2-bit number that specify a row of the S-box, and the second and third input bits specify a column of the S-box. The entry in that row and column, in base 2, is the 2-bit output.

4. The 4-bits produced by S0 and S1 undergo a further permutation P4 = [2 4 3 1], where the output of P4 is the output of the function F.

3.4. The Switch Function

The function f_k only alters the leftmost 4 bits of the input. The switch function (SW) interchanges the left and right 4 bits so that the second instance of f_k operates on a different 4 bits. In this second instance, the E/P, S0, S1, and P4 functions are the same, and the key input is K2.

4. Genetic Algorithm (GA)

GA is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a meta-heuristic) is routinely used to generate useful solutions to optimization and search problems [15].

Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover.

In a genetic algorithm, a population of candidate solution (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible [16].

The evolution usually starts from a population of randomly generated individuals, and an iterative process with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is

usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each genome is modified (recombined and possibly randomly mutated) to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced or a satisfactory fitness level has been reached for the population.

The simplest form of genetic algorithm involves three types of operators: selection, crossover and mutation. A selection operator is applied first.

Selection operator determines which chromosomes will select in the evolution process. In our study, we use Tournament selection. Two individuals are randomly selected using Tournament selection and then the highest fitness wins. Tournament selection is slightly better comparing with other selection strategies.

A crossover creates a new offspring when two parent chromosomes are mating. The simplest way to do this is choosing randomly crossover point and then everything before the selected point is copied from the first parent, while everything after a crossover point copied from the second parent. The input values for crossover point is between 0 and 1, where 0 means that a crossover point is determined randomly and 1 indicates to no crossover occurring. In our study we use a single point crossover with value of 0.65.

A mutation is performed after crossover; it used to prevent falling all solutions into a local optimum of solved problem. When mutation occurs in binary chromosomes random bit is inverted. For example, a '0' will be switched to '1' and '1' will be switched to '0'. In our work we choose a uniform mutation with 0.1 rates.

The main goal of this paper is to prove the use of GA-based method in the cryptanalysis field and to make a performance comparison between brute force search technique and GA technique. The following steps are used to break the key using GA based approach:

Step1: Determine the inputs: ciphertext and language statistics.
Step2: Generate a random pool of keys (solutions).
Step3: For each key (solution), calculate the fitness value using equation (1).
Step4: Repeating following steps to create a new population.

- Select parents (keys) according to their fitness value using Tournament selection.
- Apply Crossover on the parents to produce new offspring using a uniform crossover.
- For each offspring, apply a mutation operation to generate new offspring.
- Place a new offspring in the new population.
- Step5: Apply a new generated population for a further run of the GA.

Step6: If the end result is satisfied, stop and return the solution in current population.

5. Cost fitness

Equation (1) is the n-Grams cost function that used to find the key suitability. [6] By applying n-Grams equation the frequency for particular language can be determined by using a set of training texts and counting the occurrences of each n-Gram within the texts. The frequency of the n-Grams within a large enough sample of ciphertext will be similar to the frequency deduced from the training texts.

$$C^k = \alpha \sum(i\, \varepsilon\, \widetilde{\mathrm{A}}) \left|\mathrm{K}(i)^u - D(i)^u\right| + \beta \sum(i,j\, \varepsilon\, \widetilde{\mathrm{A}}) \left|\mathrm{K}(i,j)^b - D(i,j)^b\right| + \gamma \sum(i,j,k\, \varepsilon\, \widetilde{\mathrm{A}}) \left|\mathrm{K}(i,j,k)^t - D(i,j,k)^t\right| \quad \ldots\ldots (1)$$

Where $\tilde{A}$ denotes the English language alphabet includes the spaces between words, K and D represent known language statistics and decrypted message statistics, respectively, and u, b and t denote the unigram, diagram and trigram statistics

6. Results and Experimental Setup

In this section number of experiments is carried out to outline the effectiveness of GA by comparing the results obtained from GA with that obtained from brute force search algorithm for cryptanalysis of S-DES algorithm. The suitability of obtained key was determined using equation (1) with different key inputs.

For GA there are number of different parameters that play a vital role in the optimization process. Table 1 shows the GA parameters that used in the experiments.

Table 1. GA Parameters Used in our Experiments

Crossover	Single point
Selection	Tournament selection
Mutation	Uniform (0.1)
Number of Generation	55
Population Size	40

Table 2. Best Fitness Values for Different Sizes of Ciphertexts

Ciphertext Size (bits)	Best Fitness Value
100	0.44
200	0.43
400	0.41
600	0.43
800	0.42

Table 2 depicts the results of best fitness values after applying GA on different sizes of ciphertexts using the pool of generated keys. The best fitness value has achieved maximum number of matching between S-DES key and the key that obtained after applying GA attack.

Table 3 basically compares number of bits matching with the target key and the computational (Elapsed) time that taken to recover the keys from search space for proposed GA, brute force search and random search algorithm. Hence, Figure 2 shows the relation between ciphertext size and number of bits matched for GA technique, brute force search and random search. Comparing the computational time of these algorithms, we found that GA is not sensitive to the amount of cipher text. Figure 3 clearly shows the difference in the computational time between GA and brute force search is bigger than the difference time between GA and random search algorithm. Therefore, GA and random search algorithm are much more efficient algorithms than brute force search algorithm as almost same keys are achieved in shorter time.

Table 3. Comparison of Genetic, Random Search and Brute Force Search Algorithms

Ciphertext Size (bits)	No. of matched bits using GA	No. of matched bits using Random search	No. of matched bits using Brute Force	Elapsed Time by GA (sec)	Elapsed Time by Random search (sec)	Elapsed Time by Brute Force search

			search			(sec)
100	8	9	7	0.99	15.33	65.02
200	9	9	6	0.97	12.01	66.31
400	8	10	8	0.97	11.90	69.20
800	10	8	8	1.01	20.64	69.55
1600	9	10	7	1.20	21.00	69.70

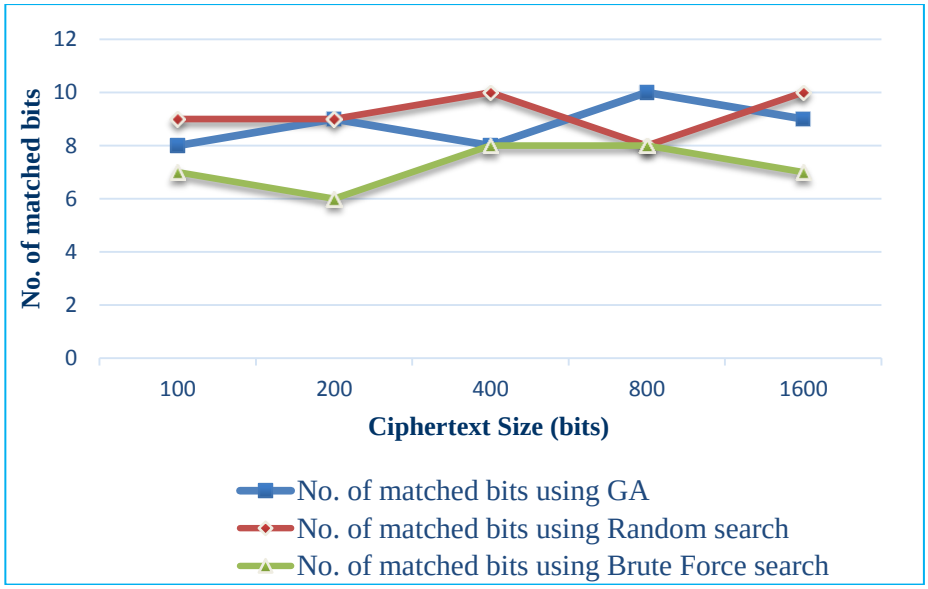

Figure 1. Comparison of Genetic Algorithm, Random Search and Brute Force Search Algorithms

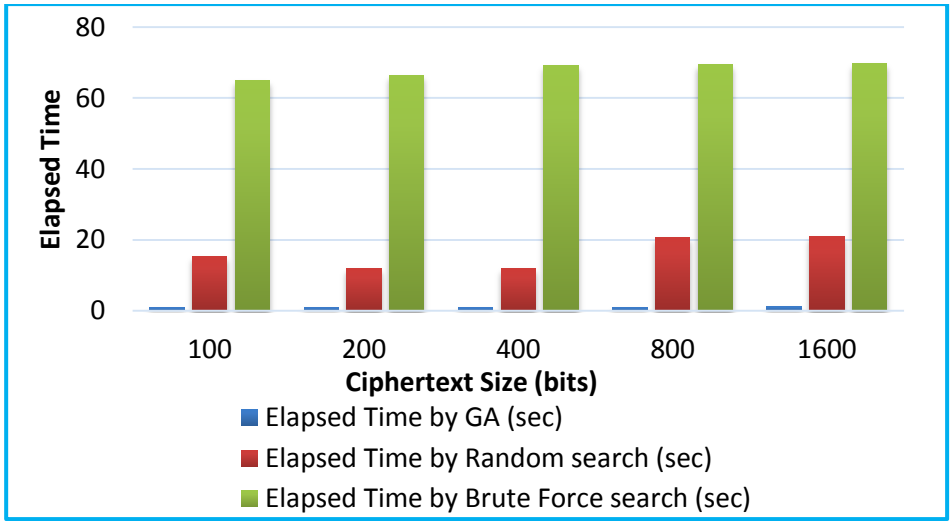

Figure 3. The Running Time Comparison of Genetic Algorithm, Random Search and Brute Force Search Algorithm

7. Conclusion

In this paper, we have presented a GA, random search and brute force search for the cryptanalysis of simplified data encryption standard algorithm. We have emphasized the importance of having a relevant fitness function when meta-heuristic methods and genetic algorithms are used to solve such problems. Hence, we have proved that the performance of GA and random search is far better than brute force search. The comparison was made on number of keys elements (out of 10) correctly recovered. Also, the computational time for recovering the keys from search space was found for GA, random search and brute force search techniques. Experiment results indicate that GA and random search algorithm are powerful techniques for the cryptanalysis of S-DES. The proposed GA can be adopted to handle other complex problems like AES and DES.

References

[1] J. H. Holland, "Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence", U Michigan Press, **(1975)**.
[2] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning", Machine learning 3.2 **(1988)**: 95-99.
[3] Teytaud, F. and C. Fonlupt, "A Critical Reassessment of Evolutionary Algorithms on the cryptanalysis of the simplified data encryption standard algorithm", arXiv preprint arXiv:1407.1993 **(2014)**.
[4] R. Vimalathithan and R. Varamarthi, 'Cryptanalysis of DES using Computational Intelligence', European Journal of Scientific Research, **(2011),**Vol. No. 55, Issue No. 2, pp. 237 – 244.
[5] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede and J. Vandewalle, ' Machine Learning in Side-Channel Analysis: A First Study' , Journal of Cryptographic Engineering, Springer- Verlag (2011), Vol No. 1, Issue No. 4, pp. 293 – 302.
[6] A. Agarwal, 'Secret Key Encryption Algorithm Using Genetic Algorithm', IJARCSSE, (2012), Vol. 2, Issue 4.
[7] L. Tao, J. Li, and J. Zhang, 'A Cryptanalysis Method based on Niche Genetic Algorithm', (2014), Appl. Math. Inf. Sci. 8, No. 1, pp.279-285.
[8] R. Vimalathithan, and M. L. Valarmathi, "Cryptanalysis of simplified-DES using computational intelligence", WSEAS Transactions on Computers 10, no. 7, 210-219, (2011).
[9] P. Garg, S. Varshney and M. Bhardwaj, "Cryptanalysis of Simplified Data Encryption Standard Using Genetic Algorithm", American Journal of Networks and Communications, 4(3): 32-36, (2015).
[10] L. Sharma, B. K. Pathak, and R. G. Sharma, "Breaking of simplified data encryption standard using genetic algorithm." Global Journal of Computer Science and Technology 12, no. 5 (2012).
[11] W. Stallings, "Cryptography and Network Security", Fifth Edition, Prentice Hall (2010), ISBN-10:0136097049.
[12] W. Stallings, "Cryptography and Network Security Principles and Practices", Third Edition, Pearson Education Inc., (2003).
[13] E. F. Schaefer, "A simplified data encryption standard algorithm" Cryptologia, 20.1 (1996): 77-84.
[14] P. Garg, "A Comparison between Memetic algorithm and Genetic algorithm for the cryptanalysis of Simplified Data Encryption Standard algorithm", arXiv preprint arXiv:1004.0574 (2010).
[15] M. Mitchell, "An introduction to genetic algorithms", PHI Pvt. Ltd., New Delhi (1996).
[16] D. Whitley, "A genetic algorithm tutorial", Statistics and computing, 4.2 (1994): 65-85.

Authors

Farah Al Adwan, she received her M.S. degree in Computer Science in 2014 from the University of Jordan, Jordan. Also, she received her B.Sc. degree in Computer Science in 2011 from Al-Balqa'a University, Jordan. Her research interests include Genetic algorithms, medical image processing and digital image processing.

Mohammad Alshraideh, he is an Associate Professor of Software Engineering in the Department of Computer Science at the University of Jordan, Jordan. He received his B.Sc. degree in Computer Science in 1988 from Mu'tah University, Jordan. He also earned from the University of Jordan, Master degree in Computer Science in 2000. Furthermore, he obtained his Ph.D. degree in Computer Science from University of Hull, UK, in 2007. During his graduate studies, he obtained a fellowship from the University of Jordan. Now he currently is working as Registrar General at the University of Jordan. His research interests include Software Testing, cost models, software engineering environments, and knowledge-based software engineering, Artificial Intelligence.

Engineer Mohammad Rasol Al Saidat, he holds a BA in Computer Engineering from Mutah University (2005), a Mater degree in Computer Engineering and Networks from the University of Jordan (2015). He is currently a data network engineering expert in Jordan Telecom Group (ORANGE) and has more than 10 years' experience in Telecommunication and Computer Engineering. His research interests include Computer Networks, Computer Security, Machine Learning algorithms and applications, Software Defined Networking, Voice over IP, IMS, and telecommunication protocols of smart systems. He holds technical certificates from Cisco, Juniper, Microsoft, ALU, RedHat, and Sonus vendors.