Genetic Algorithm Attack on Simplified Data Encryption Standard Algorithm

Poonam Garg

Institute of Management Technology
poonam@imt.edu

Abstract. With the exponential growth of networked & electronic systems, the demand of efficient and fool proof internet security is increasing. Security has emerged as a critical concern in wide range of electronic system. Cryptology is at the heart proving these securities. It consists of two complementary fields of study: cryptography and cryptanalysis. Cryptanalysis is one of the major challenging areas of intense research in the discipline of security. In this paper, we explored the use of genetic algorithm to break a simplified data encryption standard algorithm (SDES). To test its performance, we compared the implemented genetic algorithm attack with brute force search algorithm attack. Through extensive experiments and analysis it can be concluded that 1) genetic algorithms attack run ten times faster than brute force search algorithm attack with accuracy, and 2) genetic algorithms attack are 13% more accurate than brute force search algorithm attack, with same running time. A generalized version of cryptanalysis of SDES will give better insight into the attack of DES and other cipher.

Keywords: Simplified data encryption standard, Genetic Algorithm, Key search, brute force search algorithm

1 Introduction

The demand for effective internet security is increasing exponentially day by day. Businesses have an obligation to protect sensitive data from loss or theft. Such sensitive data can be potentially damaging if it is altered, destroyed, or if it falls into the wrong hands. So they need to develop a scheme that guarantees to protect the information from the attacker. Cryptology is at the heart of providing such guarantee. Cryptology is the science of building and analyzing different encryption and decryption methods. Cryptology consists of two subfields; cryptography & cryptanalysis. Cryptography is the science of building new powerful and efficient encryption and decryption methods. It deals with the techniques for conveying information securely. The basic aim of cryptography is to allow the intended recipients of a message to receive the message properly while preventing eavesdroppers from understanding the message. Cryptanalysis is the process of recovering the plaintext and/or key from a cipher. In other words cryptanalysis can be described as the process of searching for flaws or oversights in the design of ciphers.

Advances in Computer Science and Engineering.
Research in Computing Science 23, 2006, pp. 139-149

Received 01/08/06
Accepted 03/10/06
Final version 17/10/06

This paper proposes the cryptanalysis of simplified encryption standard algorithm using genetic algorithm. The cryptanalysis of simplified data encryption standard can be formulated as NP-Hard combinatorial problem. Solving such problems requires effort (e.g., time and/or memory requirement) which increases with the size of the problem. Techniques for solving combinatorial problems fall into two broad groups – traditional optimization techniques (*exact* algorithms) and non traditional optimization techniques (*approximate* algorithms). A traditional optimization technique guarantees that the optimal solution to the problem will be found. The traditional optimization techniques like branch and bound, simplex method, brute force search algorithm etc methodology is very inefficient for solving combinatorial problem because of their prohibitive complexity (time and memory requirement). Non traditional optimization techniques are employed in an attempt to find an adequate solution to the problem. A non traditional optimization technique - Genetic algorithm, simulated annealing and tabu search were developed to provide a robust and efficient methodology for cryptanalysis. The aim of these techniques to find sufficient "good" solution efficiently with the characteristics of the problem, instead of the global optimum solution, and thus it also provides attractive alternative for the large scale applications. These nontraditional optimization techniques demonstrate good potential when applied in the field of cryptanalysis and few relevant studies have been recently reported.

In 1993 Spillman [12] for the first time presented a genetic algorithm approach for the cryptanalysis of substitution cipher using genetic algorithm. He has explored the possibility of random type search to discover the key (or key space) for a simple substitution cipher. In the same year Mathew [10] used an order based genetic algorithm for cryptanalysis of a transposition cipher. In 1993, Spillman [13], also successfully applied a genetic algorithm approach for the cryptanalysts of a knapsack cipher. It is based on the application of a directed random search algorithm called a genetic algorithm. It is shown that such a algorithm could be used to easily compromise even high density knapsack ciphers. In 1997 Kolodziejczyk [8] presented the application of genetic algorithm in cryptanalysis of knapsack cipher .In 1999 Yaseen [14] presented a genetic algorithm for the cryptanalysis of Chor-Rivest knapsack public key cryptosystem. In this paper he developed a genetic algorithm as a method for Cryptanalyzing the Chor-Rivest knapsack PKC. In 2003 Grundlingh [7] presented an attack on the simple cryptographic cipher using genetic algorithm. In 2005 Garg [2] has carried out interesting studies on the use of genetic algorithm & tabu search for the cryptanalysis of mono alphabetic substitution cipher. In 2006 Garg [3] applied an attack on transposition cipher using genetic algorithm, tabu Search & simulated annealing. In 2006 Garg [4] studied that the efficiency of genetic algorithm attack on knapsack cipher can be improved with variation of initial entry parameters.

The SDES [11] encryption algorithm takes an 8-bit block of plaintext and a 10-bit key as input and produces an 8-bit block of cipher text as output. The decryption algorithm takes an 8-bit block of ciphertext and the same 10-bit key used as input to produce the original 8-bit block of plaintext. The encryption algorithm involves five functions; an initial permutation (IP), a complex function called f_K which involves both permutation and substitution operations and depends on a key input; a simple

permutation function that switches (SW) the two halves of the data; the function f_K again, and a permutation function that is the inverse of the initial permutation (IP^{-1}). The function f_K takes as input the data passing through the encryption algorithm and an 8-bit key. Consider a 10-bit key from which two 8-bit sub keys are generated. In this case, the key is first subjected to a permutation P10= [3 5 2 7 4 10 1 9 8 6], then a shift operation is performed. The numbers in the array represent the value of that bit in the original 10-bit key. The output of the shift operation then passes through a permutation function that produces an 8-bit output P8=[6 3 7 4 8 5 10 9] for the first sub key (K1). The output of the shift operation also feeds into another shift and another instance of P8 to produce subkey K2. In the second all bit strings, the leftmost position corresponds to the first bit. The block schematic of the SDES algorithm is shown in Figure 1.

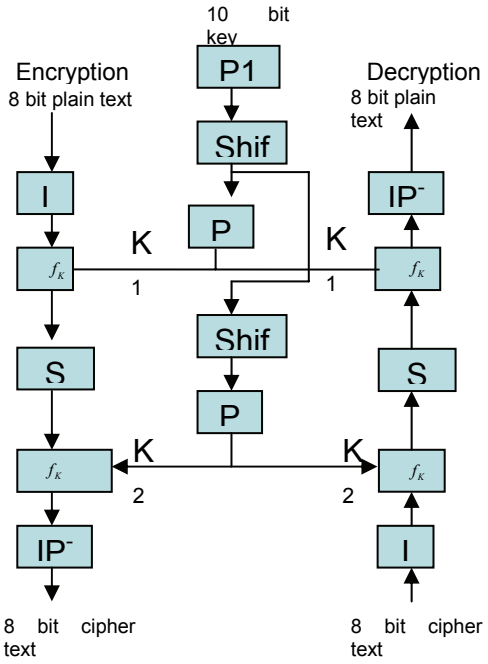

Figure 1. Simplified Data encryption scheme

Encryption involves the sequential application of five functions:

1. Initial and final permutation (IP).

The input to the algorithm is 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 applied; the

inverse permutation is done by applying, IP^{-1} = [4 1 3 5 7 2 8 6] where we have IP^{-1} (IP(X)) =X.

2. The function f_K, which is the complex component of SDES, consists of a combination of permutation and substitution functions. The functions are given as follows.
   Let L, R be the left 4-bits and right 4-bits of the input, then, f_K (L, R) = (L XOR f(R, key), R)
   where XOR is the exclusive-OR operation and key is a sub - key. Computation of f(R, key) is done as follows.
   1. Apply expansion/permutation E/P= [4 1 2 3 2 3 4 1] to input 4-bits.
   2. Add the 8-bit key (XOR).
   3. Pass the left 4-bits through S-Box S_0 and the right 4-bits through S-Box S_1.
   4. Apply permutation P4 = [2 4 3 1].

The two S-boxes are defined as follows:

$$S_0 \qquad\qquad S_1$$

$$\begin{pmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{pmatrix}$$

The S-boxes operate as follows: The first and fourth input bits are treated as 2-bit numbers 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.

3. Since the function f_K allows only 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 different 4- bits. In this second instance, the E/P, S_0, S_1 and P4 functions are the same as above but the key input is K2.

2 Objective of the Study

Cryptanalytic attack on SDES belongs to the class of NP-hard problem. Due to the constrained nature of the problem, this paper is looking for a new solution that improves the robustness against cryptanalytic attack with high effectiveness.

The objective of the study is:

- To determine the efficiency and accuracy of genetic algorithm on the cryptanalysis of SDES.
- To compare the relative performance of genetic algorithm with brute force search algorithm.

3 Genetic Algorithm

Genetic algorithms are considered as one of the most efficient search techniques. Although they do not offer an optimal solution, their ability to reach a suitable solution in considerably short time gives them their respectable role in many AI and searching techniques. The following section introduces genetic algorithms and describes their characteristics.

3.1 Genetic Algorithms Description

The genetic algorithm is based upon Darwinian evolution theory. The genetic algorithm is modeled on a relatively simple interpretation of the evolutionary process; however, it has proven to a reliable and powerful optimization technique in a wide variety of applications. Holland [9] in 1975 was first proposed the use of genetic algorithms for problem solving. Goldberg and Dejong [5] were also pioneers in the area of applying genetic processes to optimization. Over the past twenty years numerous application and adaptation of genetic algorithms have appeared in the literature. During each iteration of the algorithm, the processes of selection, reproduction and mutation each take place in order to produce the next generation of solution. Genetic Algorithm begins with a randomly selected population of chromosomes represented by strings. The GA uses the current population of strings to create a new population such that the strings in the new generation are on average better than those in current population (the selection depends on their fitness value). Three processes which have a parallel in biological genetics are used to make the transition from one population to the next (selection, crossover, and mutation) see Figure 2

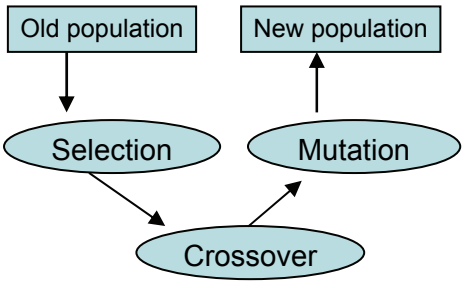

Figure 2. The basic genetic algorithm cycle

The selection process determines which string in the current will be used to create the next generation. The crossover process determines the actual form of the string in the next generation. Here two of the selected parents are paired. A fixed small mutation probability is set at the start of the algorithm.

4 Methodology: Genetic Algorithm Attack on SDES Algorithm

4.1 Problem Formulation

The possibility of using the genetic algorithm in key search is very attractive due to the ability of genetic algorithm to reduce the complexity of the search problem, and since the cryptanalysis problem is a search problem in principle, the security of a cipher is based in many cases on the complexity of the type of attacks to this particular cipher system.

An attack on the SDES using Genetic Algorithm is presented here. Three problems arises in the use of GAs in cryptanalysis

- Key representation
- Fitness function
- Stopping criteria of genetic algorithm search

4.1.1 Key Representation

The binary string is used to represent a chromosome, as key is also a binary word. The mating, crossover and mutation processes are applied directly on the candidate keys to generate new keys directed to the correct key.

4.1.2 Fitness Function

The ability of directing the random search process of the genetic algorithm by selecting the fittest chromosomes among the population is the main characteristic of the algorithm. So the fitness function is the main factor of the algorithm. The choice of fitness measure depends entirely on the language characteristics must be known. The technique used to compare candidate key is to compare n-gram statistics of the decrypted message with those of the language (which are assumed known). Equation 1 is a general formula used to determine the suitability of a proposed key(k), here ,K is known as language Statistics i.e for English, [A,…….,Z_], D is the decrypted message statistics, and u/b/t are the unigram, bigram and trigram statistics. The values of α, β and γ allow assigning of different weights to each of the three n-gram types where $\alpha + \beta + \gamma = 1$.

$$C_k \approx \alpha.\sum_{i\in A} K^u_{(i)} - D^u_{(i)} \mid + \beta.\sum_{i,j\in A} K^b_{(i,j)} - D^b_{(i,j)} \mid + \gamma.\sum_{i,j,k\in A} K^t_{(i,j,k)} - D^t_{(i,j,k)} \tag{1}$$

When trigram statistics are used, the complexity of equation (1) is O(P3) where P is the alphabet size. So it is an expensive task to calculate the trigram statistics. Hence

we will use assessment function based on bigram statistics only. Equation 1 is used as fitness function for genetic algorithm attack. The known language statistics are available in the literature [12, 13].

4.1.3 Stopping Criteria

Two variants of stop conditions are applied. In the variant-I, the algorithm will stops when the fitness function reaches to the value 1. In the variant - II the algorithm will stop either the fitness function reaches to the value 1 or generates 200 populations.

4.2 Proposed Genetic Algorithm Framework

A description of a series of test rounds is as follows.

1. Run step 2 - 11 for the S-DES.
2. Randomly select an encryption key from the main solution pool and encrypt the known plaintext message.
3. Create a new solution pool from the pool of common syllables and calculate the fitness value using equation 1 for all the solution in the new solution pool by decrypting the known-cipher text with each solution key by calculating the number of character matches in the subkey.
4. Choose two solutions with the best fitness value. Each solution should minimally able to recover at least 50% of the original plaintext message.
5. Randomly select a "crossover" point and swap the content between the two solution key arrays.
6. Evaluate the fitness for each new child (solution key) by decrypting the known-ciphertext with each "child" key and calculating the number of character matches in the subkey.
7. Choose the "child" solution with the highest fitness value.
8. Randomly select locations and mutate the selected locations with arbitrarily chosen unigram, bigram and trigram from the pool of common syllables.
9. Evaluate the fitness of the solution. If the fitness value is better then the current fitness value then update the current fitness value and the best solution variable.
10. Repeat step 8-9 until the fitness value is 1 or there is no change is best fitness for a predetermined number of iterations.
11. Repeat step 2-10 twice to produce a test series of 10 test rounds. Obtain the average number of search keys required to decrypt the full message correctly.

4.3 Implementation of the Attack

The above mentioned point was taken into consideration and implemented the attack using "C" language. The attack is implemented by generating independent two 8 bit sub-keys to represent the target key. The first generation is generated randomly using a simple uniform random number generator, to cover the bit space of the last round sub key. Then for each chromosome, a candidate sub key, the fitness is set to zero, then the entire test pair are used and the output difference is compared to the expected

differential. If the correct difference is found the fitness value is incremented and finally normalized to the number of pair.

The genetic algorithm then goes in the normal way to generate new generations. The roulette wheel is used as a selection method. The algorithm is stopped based on the criteria described earlier. The algorithm has been implemented to get key bits of the last round; essentially the attack shall continue upward to get the entire key bits.

5 Experimental Results

Experimental results obtained from genetic algorithm was generated with 100 runs per data point using 'C' language e.g. ten different messages ere created for genetic algorithm and each attack was run 10 times per message. The best result for each message was averaged to produce data point. The attack had been implemented several times, initial genetic algorithm parameters were used in the experiment are listed in Table 1.

Table1. Parameters of genetic algorithm

Population size in each generation	200
Probability of crossover	.8
Probability of mutation	.02
Minimum number of generations	100
Convergence measure	.99

To test the performance of genetic algorithm for the attack of S-DES, we compared it with brute-force search algorithm. Brute-force search algorithm tries every possibility; it always finds the best correlation. By comparing with brute-force search algorithm, we want to know how often genetic algorithm finds the optimal solution and how well it performs in average. The correlation coefficient is used as the measure of finding number of bits matched in key with accuracy.

We tried the different amount of cipher text in this experiment. Both algorithms were run on different amount of cipher text. As we are increasing the amount of cipher text, we essentially increase the search space of genetic algorithm. This enables us to track how well genetic algorithm works in different search space sizes.Table 2 list the result for genetic algorithm and brute force search algorithm, when the different amount of known cipher text is used.

By comparing the correlation column for the two algorithms, we can also see that the genetic algorithm performs better then brute force algorithms. The accuracy of finding number of bits matched in key (correlation) for both algorithms is very close for all the 10 messages. Even some of them are exactly the same. This can be better viewed using the graphical representation, as shown in Figure 3.

Comparing the running time of the two algorithms, we found that running time of genetic algorithm is not sensitive to the size of the search space. For different amount of cipher text, genetic algorithm converges at average period of 3 minutes. In contrast brute force search algorithm is more sensitive to amount of cipher text, as it takes up

to 30 minutes. From Figure 4, we can immediately see that the running time of brute force search algorithm is nearly ten times as much as that of genetic algorithm.

Table 2. Comparison of genetic algorithm and brute force search algorithm

Amount of Cipher text	Genetic Algorithm			Brute force search algorithm		
	TIME (M)	Correl.	Number of bit matched in the key (N)	TIME (M)	Correl.	Number of bit matched in the key (N)
100	2.62	.65	7	24.3	.5	8
200	4.5	.57	6	25.1	.53	6
300	2.13	.29	3	24.67	.31	4
400	2.35	.46	6	25.23	.43	7
500	2.52	.45	6	24.42	.41	6
600	2.07	.76	8	24.73	.77	9
700	4.07	.41	7	25.57	.53	7
800	2.82	.53	8	24.4	.49	8
900	2.53	.27	5	24.15	.16	6
1000	2.17	.37	9	24.35	.34	9

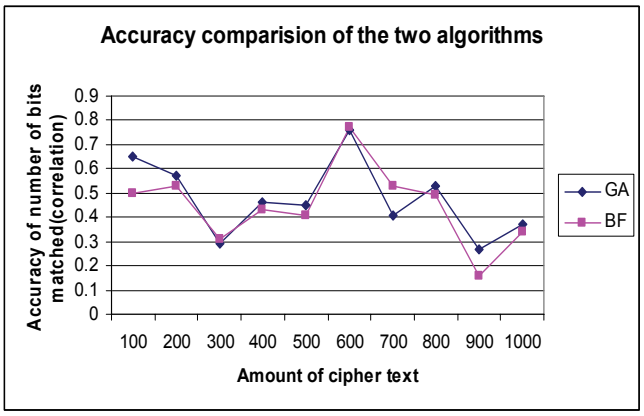

Figure 3. The accuracy of genetic algorithm and brute force search algorithm is very close. But genetic algorithm runs ten times faster (see Figure 5). The accuracy is the actual correlation found by each algorithm

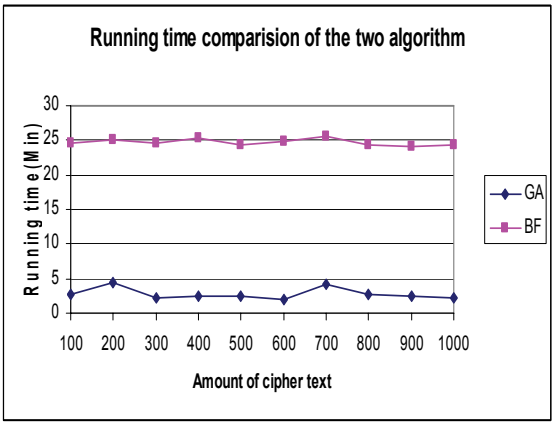

Figure 4. The running time brute force search algorithm is 10 time longer then genetic algorithm

6 Conclusion

The paper explains the genetic algorithm attack on the simplified data encryption standard algorithm. In this paper we have compared genetic algorithm and brute force search algorithm results for the attack of S-DES. Our experimental result shows that genetic algorithm is very efficient method for the cryptanalysis of S-DES. Genetic algorithm takes only 3 Minute to find the correct solution. In contrast, the brute force search algorithm takes up to 30 Minutes. When we are amount of cipher text increases, the running time of genetic algorithm keeps almost the same, but the brute force search algorithm takes more than half an hour. Genetic algorithm is an extremely effective approach for the attack S-DES. It found the best correlation for different amount of cipher text, the performance of genetic algorithm is really good. The overall average accuracy of genetic algorithm (against the best correlation found by brute-force search algorithm) is 95.04%, while average accuracy of brute force search algorithm is only 82.2%. So genetic algorithm improved the accuracy of finding the bits in the key by 13% with same running time.

References

1. Davis, L. , “Handbook of Genetic Algorithms”, Van Nostrand Reinhold, New York ,1991.
2. Garg Poonam, Sherry A.M., Genetic algorithm & tabu search attack on the monoalphabetic substitution cipher, Paradime Vol IX no.1, January-June 2005,pg 106-109

3. Garg Poonam, Shastri Aditya, Agarwal D.C, Genetic Algorithm, Tabu Search & Simulated annealing Attack on Transposition Cipher ,proceeding of Third AIMS International conference on management at IIMA – 2006, pg 983-989
4. Garg Poonam, Shastri Aditya, An improved cryptanalytic attack on knapsack cipher using genetic algorithm approach, International journal of information technology, volume 3 number 3 2006, ISSN 1305-2403, 145-152
5. Goldberg, D.E., "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, Reading, 1989.
6. Giddy J. P and Safavi-Naini R., Automated cryptanalysis of transposition ciphers, The Computer Journal, Vol 37, No. 5, 1994.
7. Gr¨undlingh, W. & van Vuuren, J. H., Using Genetic Algorithms to Break a Simple Cryptographic Cipher, Retrieved March 31, 2003 from http://dip.sun.ac.za/˜vuuren/abstracts/abstr genetic.htm, submitted 2002.
8. Kolodziejczyk, J., Miller, J., & Phillips, P. ,The application of genetic algorithm in cryptoanalysis of knapsack cipher, In Krasnoproshin, V., Soldek, J., 1997
9. Holland, J., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, 1975.
10. Methew, R.A.J. (1993, April), The use of genetic algorithms in cryptanalysis, Cryptologia, 7(4), 187-201.
11. Schaefer E, A simplified data encryption standard algorithm, Cryptologia, Vol 20, No 1, 77-84, 1996.
12. Spillman et. al., Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers, Cryptologia, 17(1):187-201, April 1993.
13. Spillman R.,Cryptanalysis of knapsack ciphers using genetic algorithms. Cryptologia, 17(4):367–377, October 1993.
14. Yaseen, I.F.T., & Sahasrabuddhe, H.V. (1999), A genetic algorithm for the Cryptanalysis of Chor-Rivest knapsack public key cryptosystem (PKC), In Proceedings of Third International Conference on Computational Intelligence and Multimedia Applications, pp. 81-85, 1999.