Cryptanalysis of Simplified Data Encryption Standard via Optimisation Heuristics

Nalini N[1], G Raghavendra Rao[2]
[1]Department of Computer Science and Engineering
Siddaganga Institute of Technology, Tumkur-572103
Karnataka, India.

[2]National Institute of Engineering, Mysore-570008
Karnataka, India.

Summary

Cryptanalysis of ciphertext has gained considerable interest among the research community engaged in security studies. Optimisation heuristics are alternative candidates for brute force attack of ciphers. This paper demonstrates the applicability of two optimisation heuristics, simulated annealing (SA) and tabu search for the cryptanalysis of Simplified Data Encryption Standard (SDES). Results of preliminary studies on a comparison with genetic algorithms (GA) are also presented.

1. INTRODUCTION

Cryptanalysis is one of the major challenging areas of intense research in the discipline of security. It is a process of looking for weakness in the design of ciphers. A cryptosystem takes as input a plaintext and a known key and produces an encrypted version of the plaintext known as the ciphertext. An attack on a cipher can be of various types. One type of attack uses the ciphertext only and attempts to arrive at the secret key and thus the plaintext. This is the most difficult attack among the classes of attacks encountered in cryptanalysis and thus we consider this type of attack in this paper.

In the brute force attack, the attacker tries every possible key on a piece of cipher text until an intelligible translation into plaintext is obtained. Cryptographic algorithms are almost designed to make a brute force attack of their solution space infeasible. The key space is large enough so that it is not possible for an attacker to try every possible key. Combinatorial optimisation techniques attempt to solve problems using techniques other than brute force. Exact and approximate algorithms can be used to solve problems from the combinatorial optimisation category. Approximate algorithms yield "good" solution to a problem. Such optimisation heuristics based on genetic algorithm [3], tabu search [2], and simulated annealing [5] have found good application in solving a large number of combinatorial optimisation problems. These techniques demonstrate good potential when applied to the domain of cryptanalysis and few relevant studies have been recently reported.

Clark [1] has carried out interesting studies on the use of optimisation heuristics for the automated cryptanalysis of classical ciphers. Simple substitution and permutation ciphers are considered in this paper. Spillman et al. [10] focus on the cryptanalysis of a simple substitution cipher. Genetic algorithm attack on the Chor -Rivest public key cryptosystem is studied by Yaseen et al. [12]. The paper by Spillman applies a genetic algorithm approach to a knapsack system [9].

Realising the lack of studies on the attack of practical cryptosystems using the optimisation heuristics mentioned above, we present in this paper our study on the cryptanalysis of Simplified Data Encryption Standard (SDES). Though it is a much-simplified version of DES, cryptanalysis of SDES using simulated annealing, genetic algorithm and tabu search will give better insight into the attack of DES and other ciphers. To the best of our knowledge, cryptanalysis of SDES using the above optimisation heuristics has not been reported earlier.

The rest of the paper is organised as follows: Section 2 presents a brief overview of the SDES algorithm. The basic principles of the relevant optimisation heuristics are presented in section 3.Experimental results are presented in section 4.Conclusions of our study are presented in section 5.

2. THE SDES ALGORITHM

The SDES [8] encryption algorithm takes an 8-bit block of plaintext and a 10-bit key as input and produces an 8-bit block of ciphertext 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 subkeys 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 the second subkey K2.In all bit strings, the leftmost position corresponds to the first bit.

The block schematic of the SDES algorithm is shown in Fig.1.

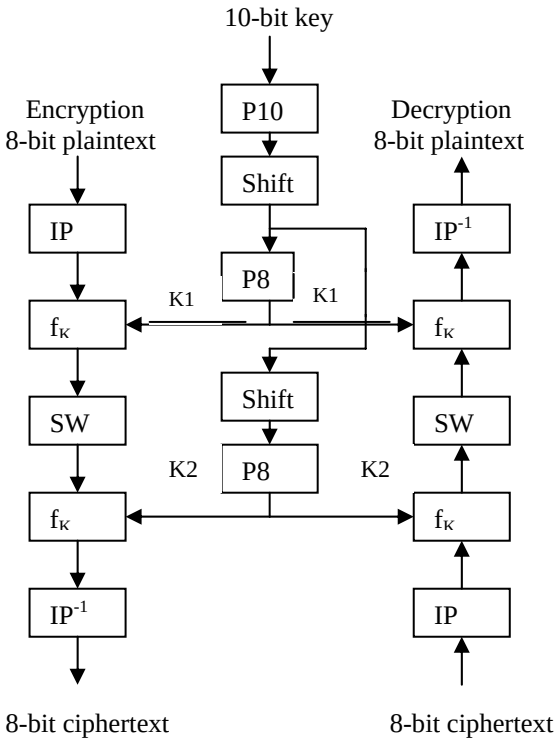

Fig. 1: Simplified DES Scheme

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

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

S_1

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

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, S0, S1 and P4 functions are the same as above but the key input is K2.

The appendix explains the computation steps of f_k through an example.

3. OPTIMIZATION HEURISTICS

Often it is hard to use exact algorithms those yield the optimal solution, due to their time or memory complexity. For many engineering applications, approximate algorithms are used to find an adequate solution to the problem. Prominent among such techniques are, simulated annealing, genetic algorithm and tabu search. These methods have a high probability of locating the global solution optimally in a multimodal search landscape. The interest in heuristics search algorithms with their inspiration from natural and physical processes began in early 1970s when Holland [4] proposed the genetic algorithms. This interest was rekindled by Kirkpatrick et al. [5] who proposed the simulated annealing technique in

1983.Simulated annealing is derived from thermodynamic considerations with annealing interpreted as an optimisation procedure. The use of tabu search was pioneered by Glover [2] from 1985 onwards, wherein the search procedure is prevented from returning to a previously explored region of the solution space too quickly. A brief overview of these techniques is presented below. Before we present the techniques, the objective function or cost function computation is discussed briefly.

A. Cost Function

An appropriate step in the formulation of the optimisation heuristics is the choice of a suitable cost function. Though few options can be considered for our problem, a natural choice is to compare the candidate keys by comparing the n-gram statistics of the decrypted message with those of the language, which are assumed to be known. Equation 1 is a general formula used to determine the cost function of a key (K), which in a way is the suitability assessment step for a key (K) [1].

$$C_K = \alpha \sum_{(i \,\varepsilon\, \tilde{A})} \left| K_{(i)}^{\,u} - D_{(i)}^{\,u} \right| + \beta \sum_{(i,j \,\varepsilon\, \tilde{A})} \left| K_{(i,j)}^{\,b} - D_{(i,j)}^{\,b} \right| + \gamma \sum_{(i,j,k \,\varepsilon\, \tilde{A})} \left| K_{(i,j,k)}^{\,t} - D_{(i,j,k)}^{\,t} \right| \quad (1)$$

In equation (1), $\tilde{A}$ denotes the language alphabet i.e., {A, B…Z, _}, for English where _ represents the space symbol), K and D denote the known language statistics and decrypted message statistics respectively, and u, b, and t denote the unigram, digram and trigram statistics respectively; α, β and γ are the weights assigning different priorities to each of the three statistics where $\alpha+\beta+\gamma = 1$. When trigram statistics are used, the complexity of equation (1) is O (P^3) where P is the alphabet size. In view of the computational complexity of trigram, only unigram and digram statistics are used. Equation (1) is used as the cost function for all our three heuristics related to the optimisation studies; minor modification of equation (1) needs to be done depending on whether the problem formulation is maximization or minimization. A minor variant of equation (1) is used as the fitness function for the genetic algorithm approach. All the following three algorithms used for cryptanalysis are presented very systematically by Clark [1]. The known language statistics are available in the literature [1] and on the web.

B. Simulated Annealing

Annealing is the process of slowly cooling a heated metal in order to attain a minimum energy state. The idea of mimicking the annealing process has been efficiently exploited by Kirkpatrick et al. [5] to solve combinatorial optimisation problems. The algorithm is initialised with a random solution to the problem being solved and a starting temperature T_0.The temperature is slowly decreased and at each temperature, a number of attempts are made to perturb the current solution. At each perturbed temperature, a change in the cost function ΔE is determined. If $\Delta E<0$, then the proposed perturbation is accepted; otherwise it is accepted with a probability indicated by the Metropolis equation given by,

$$\text{Probability } (E1 \rightarrow E2) = e^{(-\Delta E/T)} \quad (2)$$

where E1 and E2 are the cost functions, ΔE is the change in cost function and T is the current temperature. If the proposed change is accepted, then the current solution is updated. The temperature is reduced when a predefined number of attempts have been made to update the current solution. Possibilities of termination are when a certain minimum temperature is reached or a certain number of temperature reductions have occurred; or the current solution has not changed for a number of iterations. The algorithm is presented in Fig.2.

1. Input: Intercepted ciphertext, the key size P, and the language statistics.
2. Initialize the algorithm parameters: the maximum number of iterations MAX, the initial temperature T_0, and the temperature reduction factor ALPHA.
3. Set $T=T_0$ and generate a random initial solution K_{CURR} and calculate the associated cost $C_{CURR.}$
4. For I=1… MAX do
   a. Set $N_{SUCC}=0$.
   b. Repeat 100.P times
      i. Choose $n_1, n_2 \,\varepsilon[1, P]$, $n_1 \neq n_2$
      ii. Swap element n_1 with element n_2 in K_{CURR} to produce K_{NEW}.
      iii. Calculate the cost C_{NEW} of K_{NEW}. Find the cost difference $\Delta E= C_{NEW}- C_{CURR}$ and use equation (2) to determine whether the proposed transition should be accepted.
      iv. If the transition is accepted, set $K_{CURR}=K_{NEW}$ and $C_{CURR}=C_{NEW}$ and increment N_{SUCC}.If $N_{SUCC}> 10.P$, go to step 4d.
   c. If $N_{SUCC}=0$, go to step 5.
   d. Reduce T (T=T*ALPHA).
5. Output the current solution.

Fig.2: Simulated Annealing Algorithm

C. Tabu Search

The tabu search [2] prevents the search from returning to a previously explored region of the solution space too quickly. This is achieved by retaining a list of possible solutions that have been previously encountered. These solutions are called ‘tabu’; hence the name of the technique. The size of the tabu

list influences the performance of the algorithm. Tabu search is similar to simulated annealing with the added constraint of the tabu list. Two randomly chosen key elements are swapped to generate candidate solutions. In each iteration, the best new key formed replaces the worst existing one in the tabu list. The algorithm is presented in Fig.3.

1. Input: Intercepted ciphertext, the key size P, and the language statistics.
2. Initialise parameters: The size of the tabu list STABU, the size of the list of possibilities considered in each iteration SPOSS, and the maximum number of iterations MAX.
3. Initialise the tabu list with random and distinct keys and calculate the cost for each key in the tabu list.
4. For I =1,… , MAX do:
   a. Find the best key with the lowest cost in the current tabu list, $K_{BEST.}$
   b. For j=1,…, SPOSS do:
      i. apply the perturbation mechanism described in the simulated annealing attack to produce a new key $K_{NEW.}$
      ii. Check if K_{NEW} is already in the list of possibilities generated for this iteration or the tabu list. If so, return to step 4(b) i.
      iii. Add K_{NEW} to the list of possibilities for this iteration.
   c. From the list of possibilities for this iteration, find the key with the lowest cost, $P_{BEST.}$
   d. From the tabu list, find the key with the highest cost, $T_{WORST.}$
   e. While the cost of P_{BEST} is less than the cost of T_{WORST}:
      i. Replace T_{WORST} with $P_{BEST.}$
      ii. Find the new $P_{BEST.}$
      iii. Find the new $T_{WORST.}$
5. Output the best solution from the tabu list, K_{BEST}(the one with the least cost).

Fig.3: Tabu Search Algorithm

D. Genetic Algorithm

Genetic algorithms are developed based on the idea of emulating the evolution of a species. A population of individuals is generated, typically randomly. Each of these individuals represents a possible candidate solution to the problem. The solutions are encoded as bit strings (i.e., binary encoding). The solution quality of each individual is evaluated by a fitness function. In our case, the population of individuals consists of different keys considered for cryptanalysis and the fitness function is typically given by equation (1). The natural evolution process is abstracted to three genetic operations; selection, crossover and mutation. In this step, the probability of an individual to be selected is directly proportional to its fitness value. After this, the second operator, crossover, is used to create a new child out of the two selected parents by breaking up the parents' bit strings at a random position and mutually interchanging one bit string with the other. Finally, mutation is used where a randomly chosen bit in the string is flipped. Details on genetic algorithms and their application to optimisation problems are extensively treated by Goldberg [3] and Srinivas et al. [11]. An algorithmic presentation of genetic algorithm used for our study is shown in Fig.4. Keys in cryptanalysis studies are represented as a string of bits in the chromosome and genetic operators process this bit string [1], [6], [9], [10], [12].

1. Input: Intercepted ciphertext, and the language statistics.
2. Initialise the algorithm parameters: the solution pool size M and the maximum number of iterations MAX.
3. Randomly generate an initial pool of solutions P_{CURR}, and calculate the cost of each of the solutions in the pool using equation (1).
4. For I =1… MAX do:
   a. Select M/2 pairs of keys from P_{CURR} to be the parents of the new generation.
   b. Perform the mating operation on each of the pairs of parents to produce a new pool of solutions $P_{NEW.}$
   c. For each of the M children, perform a mutation operation.
   d. Calculate the cost associated with each of the keys in the new solution pool P_{NEW}.
   e. Sort P_{NEW} from the most suitable (the least cost) to the least suitable (the most cost).
   f. Merge P_{CURR} with P_{NEW} to give a list of sorted solutions (discard duplicates). Choose the best M keys to become the new current pool $P_{CURR.}$
5. Output the best solution from $P_{CURR.}$

Fig. 4: Genetic Algorithm for Cryptanalysis

The mutation operation in the algorithm of Fig.4 is identical to the solution perturbation method used in simulated annealing attack discussed earlier. That is, randomly select two elements in the child and swap those elements.

4. EXPERIMENTAL RESULTS

We have carried out extensive experimentation to arrive at the key and thus the plaintext, given the ciphertext. The attack studies have been carried out with three different types of text.
1. A typical English novel text (maximum of 3000 characters long).
2. A typical technical text.
3. An E-commerce type of text for fund transfer.

The experiments were conducted on a P4 system using Matlab version 7.1.For the cost function given in equation (1), it was noticed that the benefit of trigrams over digrams was small.

Our objective in this paper is to compare the results obtained from simulated annealing and tabu search attacks with those obtained using genetic algorithms. The results of our study on cryptanalysis of SDES using genetic algorithms are presented in a paper by Nalini et al. [7]. Due to space constraints, the details of the underlying experiments, results and discussions for the genetic algorithm-based cryptanalysis are not presented here and these can be found in the paper by Nalini et al. [7].

For the 50 cases tested, in about 48 cases the keys were obtained within 5 iterations for the tabu search attacks. However, the computation time and the corresponding number of iterations varied for different cases under consideration. Out of the three optimisation heuristics considered, the tabu search emerged to be the most effective one in terms of the percentage of successful key retrievals and the number of iterations taken. For the tabu search technique, the size of the tabu list influences the performance of the algorithm. It was noticed that a tabu list size of 15 was a good choice guaranteeing key retrieval for all the cases considered. Table 1 shows the effect of the tabu list size on the number of bits matched, out of the 10 bits in the key.

TABLE 1:EFFECT OF TABU LIST SIZE

Sl. no	Tabu list size	No.of bits matched
1	15	10
2	20	10
3	12	09
4	08	06

For the simulated annealing attack, it was noticed that in case of *successful* retrieval of the key, this attack was faster compared to tabu search and genetic algorithm attacks; in most cases, the simulated annealing attack retrieved the key in about 15 minutes, whereas the genetic algorithm and tabu search took about 20 and 10 minutes respectively. However, the simulated annealing attack needs tuning of certain parameters.

In case of simulated annealing attack, the initial temperature and the rate of temperature decrease are important parameters influencing convergence to the correct result. It was observed that as the initial temperature was increased, the number of bits matched in the key decreased. Thus a high initial temperature is not desirable.

For the temperature profile, we considered linear and exponential decay of the profile. For a linear profile, the temperature at iteration K is given by,

$$T_K=T_{K-1} - ((T_0-T_\infty)/MAX)$$

where T_0 is the initial temperature, T_∞ is the final temperature and MAX determines how much the temperature is reduced at each step. For an exponential decay of temperature, we have, $T_K= \delta T_{K-1.}$

We have studied the performance of both these temperature reduction schemes and it is concluded that the exponential decay gives better results. Initial temperature T_0 in the range 0.4-1 and δ in the range 0.02-0.1, yield good results.

For genetic algorithm attack, the key was retrieved on an average in 5 generations, each generation takes 5 minutes; thus on an average it takes about 20-25 minutes with a population size of 50 for each generation. The search space thus has a dimension of 250 compared to 1024 in the brute force case. Thus in case of genetic algorithm, the search space reduction is by a factor $1024/250 \approx 4$.For a comparison of the execution times of genetic algorithm, tabu search, and simulated annealing attacks with the execution time of the brute-force method, we notice that in the worst case the brute-force attack takes about 35-40 minutes.

Table 2 shows a snapshot of some typical parameter value combinations for which the key was successfully retrieved among the many experiments carried out by us. Many such combinations yielded success.

TABLE 2: TYPICAL PARAMETERS YIELDING CONVERGENCE USING GA

Sl. no	α, β, γ	Selection method	Crossover fraction	Mutation rate	Population size
1	0.2,0.4, 0.4	Roulette	0.8	0.05	100
2	0.2,0.5, 0.3	Stochastic	0.5	0.05	100
3	0.2,0.4, 0.4	Tournament	0.8	0.03	50

To obtain the performance measures, each of the attacks was tried a large number of times with different combinations of the parameters. The average value obtained over fifty messages is used as the performance metric. This was done primarily due to the random nature of the algorithms. We believe that averaging of results of a large number of attacks will yield a reasonably representative result of the performance statistics under consideration.

We broadly notice that all the three algorithms do not significantly differ as far as the eventual success of the attack is concerned. However, it is noticed that as the amount of known ciphertext is increased, the tabu search attack performs better than the other two attacks based on simulated annealing

and genetic algorithm. This is evident from the mean and standard deviation of the number of key bits successfully obtained, as an average over fifty runs. Tabu search shows a higher mean value with lower standard deviation (variance). Table 3 shows some relevant data based on which these inferences have been drawn.

We next consider the complexity of the algorithms viewed from the perspective of number of keys involved in a successful attack and the time required by the algorithms for different percentages of key elements of the key retrieved. From such an observation, it is noticed that simulated annealing performs better than the genetic algorithm. However from these studies it can be inferred that the simulated annealing algorithm considers more number of solutions than the genetic algorithm and in lesser time, for a successful retrieval of the key. From a consideration of the above two aspects of characterising complexity of the algorithms, the tabu search is more efficient than the other two methods.

TABLE 3:STATISTICAL DATA ON PERCENTAGE OF SUCCESSFUL ATTACKS (μ: MEAN, σ: STANDARD DEVIATION)

Amount of Cipher text	SA		GA		Tabu Search	
	μ	σ	μ	σ	μ	σ
200	7.5	2.3	7.4	2.8	8.1	2.7
500	8.4	2.0	8.1	2.5	9.2	2.2
1000	9.2	1.9	9.1	2.2	10	1.8

5.CONCLUSIONS

The paper has demonstrated that optimisation heuristics such as genetic algorithm, tabu search and simulated annealing are ideally suited for the cryptanalysis of Simplified Data Encryption Standard. Thus these techniques offer a lot of promise for attacks of other ciphers. Though SDES is a simple cipher, its building blocks are also used in other ciphers. Experimental results demonstrate good performance for tabu search and simulated annealing; few parameters need to be tuned for the best possible performance. If these parameters are tuned properly, one may get much better performance for these two methods.

REFERENCES

[1] Clark A and Dawson Ed, "Optimization Heuristics for the Automated Cryptanalysis of Classical Ciphers", Journal of Combinatorial Mathematics and Combinatorial Computing, Vol. 28,pp. 63-86, 1998.

2] Glover Fred, Taillard Eric and Werra Dominique de, "A User's Guide to Tabu Search" Annals of Operations Research, Vol. 41,pp. 3-28,1993.

[3] Goldberg D.E, "Genetic Algorithms in Search, Optimisation and Machine Learning", Boston, Addison-Wesly, 1989.

[4] Holland John H, "Adaptation in Natural and Artificial Systems", Ann Arbour MI, University of Michigan Press, 1975.

[5] Kirkpatrick S, C .D. Gelatt. Jr. and Vecchi M. P, " Optimisation by Simulated Annealing", Science, Vol. 220, No. 4598, pp. 671-680,1983.

[6] Nalini N and Raghavendra Rao G,"A New Encryption, Decryption and Message Digest Algorithm Combining The Features of Genetic Algorithm and Cryptography", In the Proceedings of EUROGEN 2005, Munich, Germany, Sept 12-14, 2005.

[7] Nalini. N and Raghavendra Rao G, "Cryptanalysis of Simplified Data Encryption Standard (SDES) using Genetic Algorithm", submitted to International Journal of Information and Computer Security (IJICS), Inderscience Publishers.

[8] Schaefer E, "A Simplified Data Encryption Standard Algorithm", Cryptologia, Vol .20, No.1, pp. 77-84, 1996.

[9] Spillman R,"Cryptanalysis of Knapsack Ciphers using Genetic Algorithms", Cryptologia, Vol.17, No.4, pp. 367-377, 1993.

[10] Spillman R, Janssen M, Nelson B and Kepner M, "Use of Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers", Cryptologia, Vol. 17, No.1, pp. 30-44.1993.

[11] Srinivas M and Patnaik L.M, "Genetic Algorithms: A Survey", IEEE Computer, pp.17-26, 1994.

[12] Yaseen I F T and Sahasrabudde H V,"A Genetic Algorithm for the Cryptanalysis of Chor-Rivest Knapsack Public-key Cryptosystem", Proceedings of Third International Conference on Computational Intelligence and Multimedia Applications, pp. 81-85, 1999.

APPENDIX

Computation of Function f_k

Plaintext: 10100101, Apply IP: 01110100

Compute f_k (01110100) for K=00101111

f_k (01110100)=((0111) XOR f (0100,K), 0100)

To compute f (0100, K) 1.Apply E/P: 00101000

2.Add K: 00101111 Sum: 00000111

3. Pass 0000 to S0, 0111 to S_1

S_0: Read row 0,column 0: 1=$(01)_2$and S_1: Read row1, column 3:3=$(11)_2$ Output of S-boxes: 0111

4. Apply P_4 =1110

. ˙ . f_k = ((0111) XOR (1110), 0100))

= 01110100

Nalini.N, received her B.E degree from University BDT College of Engineering,Davanagere,Kuvempu University,India in the year 1996,her M.S(Software Systems) degree from BITS,Pilani,Rajasthan,India in the year 1999.She is curently persuing her Ph.D from Visvesvaraya Technological University,Belgaum,India.Also she is working as an Assistant Professor in the department of CSE,Siddaganga Institute of Technology,Tumkur,India.She has presented more than six papers at various National and International Conferences.Her research interests are in the areas of Cryptography and Optimisation Heuristics.

Dr.G.RaghavendraRao, Completed his BE, ME & Ph.D from University of Mysore, Indian Institute of Science, Bangalore & University of Mysore, respectively.Has been teaching Computer Science for the last 23 years. Presently the Principal and also Head of the Department of Computer Science & Engg. at National Institute of Engineering, Mysore, India.He Has more than 25 papers in International and National Journals and Conferences.His Areas of interest include Genetic Algorithms, Cryptography, Data mining, Webcommerce and Artificial Intelligence.He is also the member of IEEE & ISTE.