Cryptanalysis of Block Ciphers via Improvised Particle Swarm Optimization and Extended Simulated Annealing Techniques

Nalini N[1] and G. Raghavendra Rao[2]

(Corresponding author: Nalini N)

Department of Computer Science and Engineering, Siddaganga Institute of Technology[1]
Tumkur-572103, Karnataka, India (Email: nalinaniranjan@hotmail.com)
National Institute of Engineering[2]
Mysore-570008, Karnataka, India. (Email: principal@nie.ac.in)

(Received Jan. 16, 2006; revised and accepted Feb. 21, 2006)

Abstract

Optimization heuristics have been pursued in recent years as a viable approach in cryptanalysis. Even in simple ciphers where brute force method is successful, use of these techniques demonstrates their potential application in attacks of complex ciphers. This paper establishes the applicability of a couple of optimization heuristics to cryptanalysis studies; one based on thermostatistical persistency applied to simulated annealing and the other one based on particle swarm principle. Though both methods lead to successful attacks, our improvised version of group swarm optimization yields better performance. As a vehicle of demonstration of our concept, we choose simple yet representative block ciphers such as computationally tractable versions of DES, for our studies.

Keywords: Cryptanalysis, DES, heuristic optimization, particle swarm optimization, simulated annealing, simplified DES

1 Introduction

Cryptanalysis is the study of methods for obtaining the meaning of encrypted information. Cryptanalysis has coevolved together with cryptography, and the context can be traced through the history of cryptography; new ciphers being designed to replace old broken designs and new cryptanalytic techniques invented to crack the improved schemes. In order to create a secure cryptosystem, it will have to be designed against possible attacks. Design of ciphers resistant to attacks needs a good exposure to their strong and weak features, which is possible by systematic cryptanalysis studies. Two fundamental goals in computer science are finding algorithms with provably good run times and with provably good or optimal solution quality. A heuristic is an algorithm that gives up one or both of these goals; for example, it usually finds pretty good solutions, but there is no proof the solutions could not get arbitrarily bad; or it usually runs reasonably quickly, but there is no argument that this will always be the case. Often, one can find specially crafted problem instances where the heuristic will in fact produce very bad results or run very slowly; however, these instances might never occur in practice because of their special structure. Therefore, the use of heuristics is very common in real world implementations.

Here we have made an attempt to use heuristic techniques in the cryptanalysis of simplified variants of DES (Data Encryption Standard). Though such simplified versions are amenable to brute force attacks, studies reported in this paper are useful in the cryptanalysis of other complex ciphers and in exploring the weakness of ciphers. The basic building blocks of most block ciphers being of similar nature, it is envisaged that our studies can be extended to study attacks of other ciphers.

Particle swarm optimization [3] has demonstrated excellent promise as a heuristic technique in recent years. We propose an extension to this concept by introducing a unique concept of group of swarms. This technique when applied to cryptanalysis of our candidate block cipher has yielded better performance compared to the simulated annealing algorithm implemented with thermostatistical persistency principle [5] incorporated into it.

The rest of the paper is organized as follows. Section 2 presents a brief overview of the DES algorithm. The principles of simulated annealing and thermostatistical persistency are explained in Sections 3 and 4, along with the experimental results. The underlying principle of particle swarm optimization (PSO) and our novel concept of group PSO and the results obtained from this method are presented in Section 5. More experimental results on the cryptanalysis of simplified DES are presented in Section

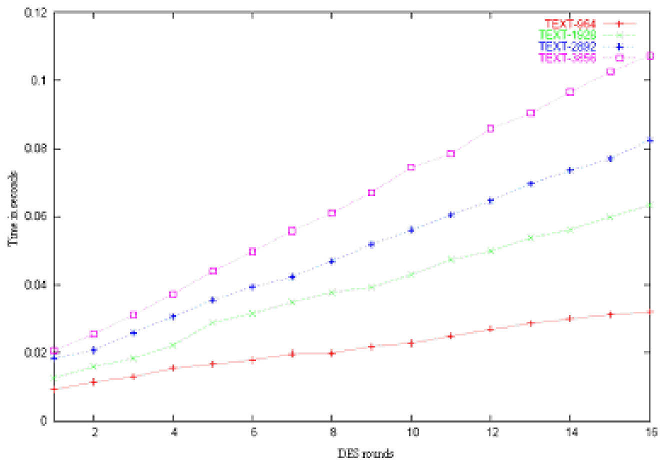

Figure 1: Performs of DES, on P4, 1.7GHz processor

6. A brief discussion on linear and differential cryptanalysis studies of such ciphers is presented in Section 7. The paper concludes with Section 8.

2 Data Encryption Standard

Data Encryption Standard (DES) was designed by IBM in 1970; later it was adopted as a standard. Here we consider a ciphertext only attack on DES using heuristic techniques. A detailed description of the algorithm is provided in [7]. We have implemented the same in C language on a P4 system. A performance of the code is given in Figure 1. Here we consider text length of 964 bytes, 1928 bytes, 2892 bytes and 3856 bytes. In each case, we recorded encryption time at each round, the decryption time being the same. For the purpose of cryptanalysis, the code is written so as to take number of rounds as input for encryption and decryption.

2.1 Simplifying the Problem

For the purpose of cryptanalysis, two cases were considered.

1) Reduced number of rounds:
   Here we considered only few rounds of DES algorithm so as to simplify the problem. Typically in cryptanalysis we used DES with four rounds.

2) Removing SBOX:
   The strength of DES lies in the non-linearity induced by SBOX, so we thought of initially eliminating the SBOX in the encryption and decryption phases. But looking into the algorithm, the elimination of SBOX is not straightforward. From Section 2.2, it is clear that each round takes 32-bit inputs L_{i-1} and R_{i-1} from the previous round, and produces 32-bit outputs

					E
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

			P
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Figure 2: Expansion and permutation

Table 1: Eliminating SBOX

1	2	3	4	5	6	9	10
11	12	15	16	17	18	21	22
23	24	27	28	29	30	33	34
35	36	39	40	41	42	45	46

L_i and R_i, for $1 \leq I \leq 16$, as follows;

$$\begin{aligned} L_i &= R_i - 1 \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i), \quad \text{where} \\ f(R_{i-1}, K_i) &= P(S(E((R_{i-1}) \oplus K_i). \end{aligned} \qquad (1)$$

From Equation (1), R_{i-1} (32-bit) is expanded to 48-bit; using the table shown in Figure 2 and it is XORed with K_i. Result of this is subjected to SBOX, to get 32-bit output. To eliminate the SBOX, we use Table 1 after XOR operation in Equation (1). We have derived the entries in the table from the expansion table shown in Figure 2, by eliminating all those entries, which occur second time.

2.2 DES Algorithms

```
INPUT: Plaintext $m1, \ldots, m64$; 64-bit key $K = k1,$
        $\ldots, k64$ (including 8 parity bits)
OUTPUT: 64-bit cipher text block $C = C1, \ldots, C64$
1. Key schedule: Compute sixteen 48-bit round keys $Ki$
   from $K$.
2. $(L0, R0) \leftarrow$ IP $(m1m2 \ldots m64)$
3. for $I$ from 1 to 16, compute $Li$ and $Ri$
   a. Expand $Ri - 1 = r1r2 \ldots r32$ from 32 to 48 bits,
      $T \leftarrow E(Ri - 1)$
   b. $T' \leftarrow T \oplus Ki$. Represent $T'$ as eight 6-bit
      character strings $(B1 \ldots B8) = T'$
   c. $T'' \leftarrow (S1(B1), S2(B2), \cdots, S8(B8))$
   d. $T''' \leftarrow P(T'')$
4. $b1b2 \ldots b64 \leftarrow (R16, L16)$(Exchange final blocks $L16$,
   $R16$)
5. $C \leftarrow IP^{-1}(b1b2 \ldots b64)$
```

3 Simulated Annealing (SA)

Simulated annealing [4] is based on the concept of annealing. In physics, the tern annealing describes the process of slowly cooling a heated metal in order to attain a "minimum energy state". A heated metal is said to be in a state of "high energy". The molecules in a metal at a sufficiently high temperature move freely with respect to each other. However when the metal is cooled, the molecules lose their thermal mobility. If the metal is cooled slowly, a "minimum energy state" is reached. To apply the analogy of annealing to the field of combinatorial optimization, it is useful to think of the slowly cooled metal as having reached a crystalline structure in which the molecules are ordered and the energy is low. This is analogous to the optimal solution to a problem which is "ordered" and represents the lowest "cost" to solve the problem being optimized, for a minimization problem. The technique merges hill-climbing with the probabilistic acceptance of non-improving moves. The search starts at some initial state $S = S_0$. There is a control parameter T known as the temperature. This starts 'high' at T_0 and is gradually lowered. At each temperature, a number of moves to new states are attempted. A candidate state is randomly selected from the neighborhood of the current state. The change in value of the cost function is calculated. If it improves the value of cost function, then a move to that state is taken; if not, then it is taken with some probability. Probabilistic acceptance is determined by generating a random value in the range $(0 \ldots 1)$ and performing the indicated comparison. The algorithm is discussed below.

In a simulated annealing attack, the key is represented as a string of N characters in the alphabet. A very simple way of perturbing such a key is to swap the key elements in two randomly chosen positions. This is the method utilized in the following algorithm which describes a simulated annealing attack on the cipher considered in this paper.

```
Algorithm: Simulated Annealing
1. Generate a solution to the problem (randomly or
   otherwise) and determine its cost.
2. Initialize the temperature $T = T0$.
3. At temperature $T$, repeat $J$ times...
  a. Perturb the current solution to give a "Candidate"
     solution.
  b. Find the cost of the candidate solution and
     determine the difference between its cost and the
     cost of the current solution.
  c. Using the cost difference ($\triangle E$) and the
     current temperature ($T$),
```

$$Pr(E1 \leftarrow E2) = e^{(-\triangle E/KT)}, k = 1. \qquad (2)$$

```
     This gives the probability that the candidate solution
     should be accepted. Generate a random number in
     the interval [0, 1]. If the random number is less
     than the probability returned by Equation (2), then
     the candidate is accepted.
  d. If the candidate is accepted, then the current
     solution and its cost are updated.
4. If the stopping criteria are satisfied, then discontinue;
   otherwise, reduce the temperature $T$ and repeat from
   Step 3.
```

We have implemented the algorithm for cryptanalysis of DES. Here we have experimented on various aspects of SA.

3.1 Fitness Function or Cost

1) Monogram and bigram statistics:
To implement this fitness function, the frequency of each character in the decrypted text is calculated. This frequency is normalized by dividing it by the total number of characters in the file. This normalized frequency is then subtracted from the expected frequency of the character in normal English text. The absolute value of this difference is taken. The differences for all characters are added together. The normalization takes care that this value always lies between 0 and 1.

The bigram is an extension of unigram to two characters. Now rather than calculating frequency of individual character, we calculate frequency of "pairs" of letters. For example, a pair "an" will always appear more frequently than pair "bt". Again statistics for the frequencies of these pairs are also available. These statistics are compared with the statistics obtained from the decrypted text.

To implement this fitness function, the frequency of each pair of letters in the decrypted text is calculated. This frequency is normalized by dividing it by the total number of pairs in the file. This normalized frequency is then subtracted from the expected frequency of the pair in normal English text. The absolute value of this difference is taken. The differences for all pairs are added together. The normalization takes care that this value always lies between 0 and 1.

The fitness function based on monogram and bigram is given by,

$$\sum\{|SF[i] - DF[i]| + \sum |SDF[i,j] - DDF[i,j]|\}/4, i = 1 \text{ to } 26, j = 1 \text{ to } 26.$$

Here the letters $A \ldots Z$ are referenced by the indices $1 \ldots 26$, $SF[i]$ is the standard frequency of character i in English, $DF[i]$ is the measured frequency of the character i in English. SDF is the standard bigram frequency and DDF is the decoded bigram frequency.

2) Count of intelligible characters represented by ASCII:
This is the most simple, efficient and effective fitness

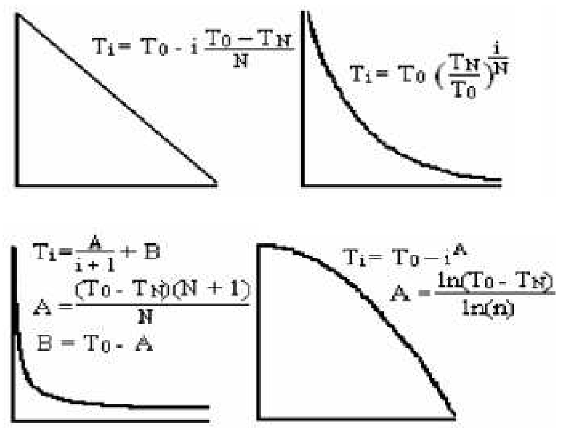

Figure 3: Coding schedules

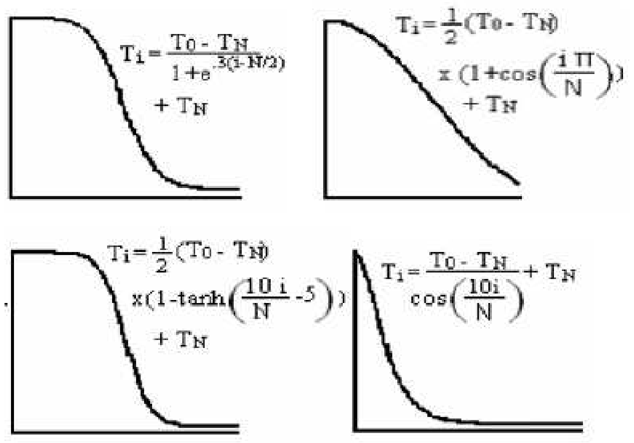

Figure 4: Cooling schedules

function of all the fitness functions tried in our experiments. To calculate fitness of a key, we simply calculate the number of readable characters in the decrypted text ($a-z$ and $A-Z$). We further normalize this value by dividing this number by the size of the file in bytes. For a perfect key, all the characters will be readable and hence this value will be one. Since our algorithms are implemented to solve minimization problems, we simply subtract this value from 1 to get the fitness of a key.

Intuition behind this approach is that the message is made up of readable characters. So the decrypted text should also contain the readable characters. More the intelligible characters, better is the key, lower is the fitness value of the key. Since this approach does not require any table lookups, it is also the most efficient approach seen so far.

In the experiments performed, it was found that this function does give fair weight to the key. The fitness value decreases more rapidly than other fitness function values. It also spans almost the complete range from 0 to 1.

The formula for this fitness function can be given as follows:

$$f = 1 - c/n,$$

where c = number of characters falling in the range $a-z$ or $A-Z$; n = total number of characters in the file.

3) Combination of different cost functions with different weights: different weight factors to monogram and bigram statistics have been assigned.

In one more approach of assigning the fitness values to individuals, we used combination of more than one fitness function and assigned weights to it. Thus the new fitness value was calculated as follows:

$$f = \alpha * \text{unigram} + \beta * \text{bigram} + \mu * \text{intelligible\_char},$$

where α, β, and μ are the weights of the respective fitness functions. This fitness function was also found to be quite useful during our cryptanalysis studies.

3.2 Cooling Schedule

Cooling schedule is a very important component of simulated annealing, it provides a way of accepting a candidate solution with higher cost. This helps in overcoming local minima. A cooling schedule should slowly decrease the temperature. Figures 3 and 4 show some typical cooling schedules those have been used with our simulated annealing optimization, though many such profiles have been studied by us. Ti is the temperature for cycle i, where i increases from 0 to N. The initial and final temperatures, $T0$ and TN respectively, are determined by the user as is N. The last temperature profile under Figure 4 yields better results in terms of convergence, compared to other temperature profiles.

3.3 Experimental Results

Experimental results are shown in Figures 5 and 6. It is found that 4 round DES, even without SBOX, seems to be strong enough against ciphertext only attack using simulated annealing. This motivated us to explore other heuristics.

4 Thermostatistical Persistency

SA is very inefficient at low temperature when it spends most of the time refusing proposed transitions. Due to its memoryless nature, it does not provide any means of restricting the search space. However, the character of the method changes from free search to quite specific minimization.

In [2] Chardaire et al. introduced a method called "Thermostatistical Persistency (TP)" to strengthen the simulated annealing optimization technique. Basically, it

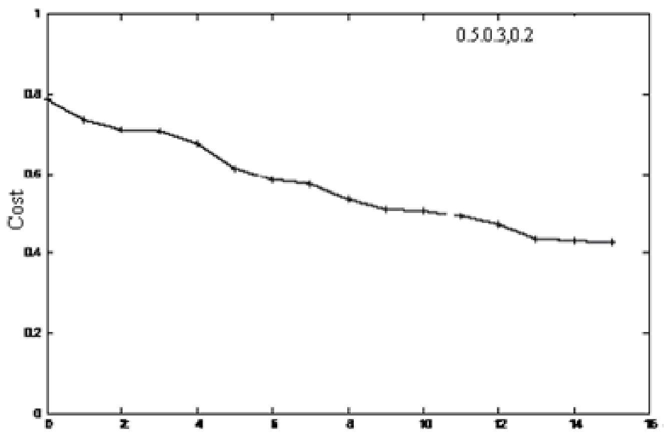

Figure 5: Simulated annealing for cryptanalysis of 4 round of DES without Sbox, with combination of fitness functions with weights 0.5, 0.3, 0.2 for intelligent characters, mono and bigram resply. Initial cost=0.786687, cost at saturation=0.423511, initial temperature=20, temperature at saturation=2.2222, time taken=93 hrs, length of text=3856 bytes.

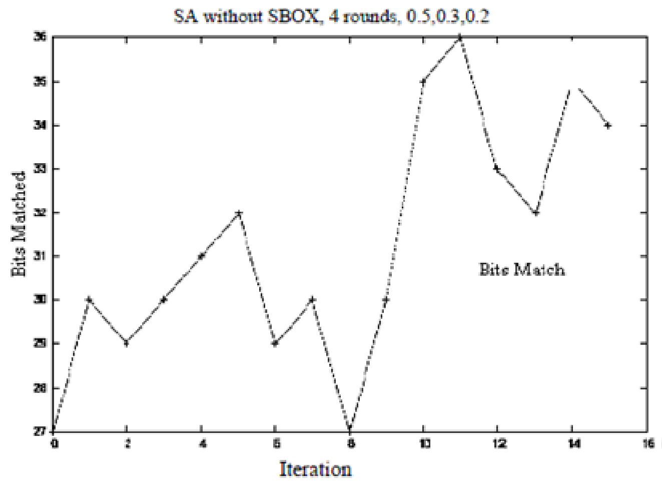

Figure 6: Number of bits matched with actual key during experiment

aims at progressively limiting the configuration space during the simulated annealing procedure. At a given temperature, mean value of every bit being 1 or 0 is calculated. If it is greater than the threshold value, then the bit is frozen assuming that with high probability it is tending towards either 0 or 1. We have studied the cryptanalysis of DES, with four rounds and SBOX, using thermostatistical persistency. The results are shown in Figures 7 and 8. Following are some important observations:

1) Keeping SBOX in DES makes it difficult for cryptanalysis.

2) From a comparison of the performance of different fitness functions, we found that for this problem number of intelligible characters as a fitness function seems to be better in terms of performance.

3) The number of bits matched in the actual key and the number of bits frozen appear to be coming close to each other (Figure 8). But it is found that, some bits frozen do not match with the actual key bits, thus creating deadlock. In all cases, the output is recorded after giving sufficient time for the process to saturate.

5 Particle Swarm Optimization

In [3] Kennedy et al. propose a new technique called Particle Swarm Optimization (PSO), which borrows the idea from bird flocking, fish schooling, and swarm theory. We have used this technique in the cryptanalysis of DES. The results are shown in Figure 9.

5.1 The Principle

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. PSO has been successfully applied in many areas; function optimization, artificial neural network training and fuzzy system control. PSO has been used in cryptography to solve the tough problem of integer factorization.

PSO simulates the behavior of bird flocking. Suppose a group of birds are randomly searching food in an area. All the birds do not know where the food is. The effective strategy is to follow the bird which is nearest to the food. In PSO, each single solution is a "bird" in the search space. We call it "particle". All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of

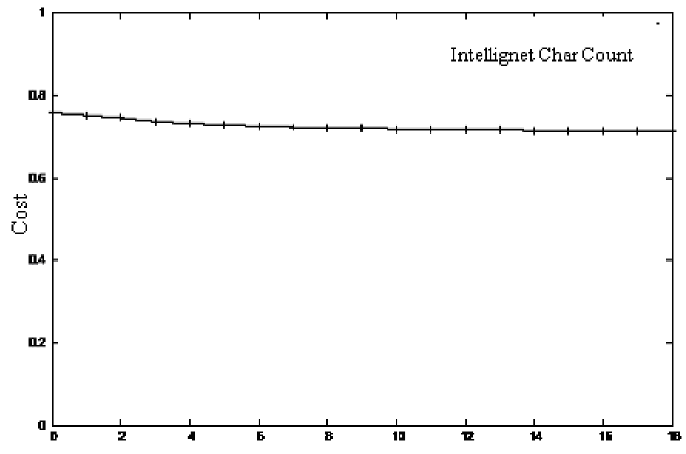

Figure 7: Thermostatistical persistency for cryptanalysis of 4 round DES with Sbox Fitness function is INTELLIGENT CHARACTER COUNT. Initial cost=0.758367, cost at saturation=0.712471, initial temperature=20, temperature at saturation=8.333, time taken=46 hrs. Length of text=3956 bytes.

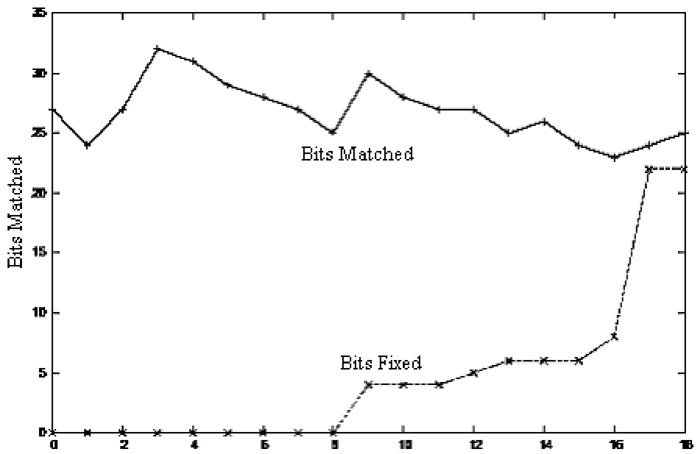

Figure 8: Number of bits matched with actual key and number of bits frozen

the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In each iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. This value is called pbest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and is called gbest. When a particle takes part of the population as its topological neighbors, the best value is a local best and is called lbest.

After finding the two best values, the particle updates its velocity and position with the following equations:

$$\begin{aligned} v[\cdot] &= v[\cdot] + p\_incr * rand() * (pbest[\cdot] - present[\cdot]) \\ &\quad + g\_incr * rand() * (gbest[\cdot] - present[\cdot]) \\ present[\cdot] &= persent[\cdot] + v[\cdot], \end{aligned} \tag{3}$$

where, $v[\cdot]$ is the particle velocity, $persent[\cdot]$ is the current particle (solution), $pbest[\cdot]$ and $gbest[\cdot]$ are defined as stated before, rand() is a random number between (0,1), *p_incr*, *g_incr* are learning factors. Usually *p_incr*, *g_incr* = 2. However these values of *p_incr*, *g_incr* are problem-dependent. As it can be easily seen, higher values of *g_incr* help particles to move out of local minima. Thus, usually in our experiments, we have taken higher values of *g_incr*.

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user, then the velocity on that dimension is limited to Vmax.

In our implementation, each particle corresponds to a key. The fitness of a key is found by one of the fitness functions discussed in the previous sections. Since DES uses 7 - byte key, our search space is 7-dimensional. Thus each byte corresponds to a separate direction. Vmax is varied throughout the experiments. The range varied is from 8-64. Remember that, a particle can take a maximum of 256 different values in each direction.

5.2 *p_incr* and *g_incr*

As we have seen, two parameters we can play around in PSO are *p_incr* and *g_incr*. They are so called learning factors of the algorithm. Higher values of *p_incr* allows a particle to move towards its direction of search faster. Higher values of *g_incr* allow all the particles to move in the direction of group leader faster. These are very essential parameters in PSO. In a discrete optimization problem such as cryptanalysis of DES, it is better to have higher values of *g_incr*.

5.3 Algorithm

```
Input: Number of particles in the swarm, p_incr, g_incr,
       pointer to fitness function.
Output: The key having the lowest fitness as found by
        PSO.
Step 1: Generate particles randomly to form a swarm.
Step 2: Calculate the fitness function of each of the
        particles. If the current position of the particle
        is better than the previous history, update the
        particle's history to indicate this fact.
Step 3: Find out the best particle of the swarm. Update
        the positions of the particles by Equation (3).
Step 4: If the maximum number of iterations has exceeded
        or if the key with very low fitness value is found,
        then go to Step 5 or else go to Step 2.
Step 5: Copy the best key obtained so far in the output
        key variable and exit.
```

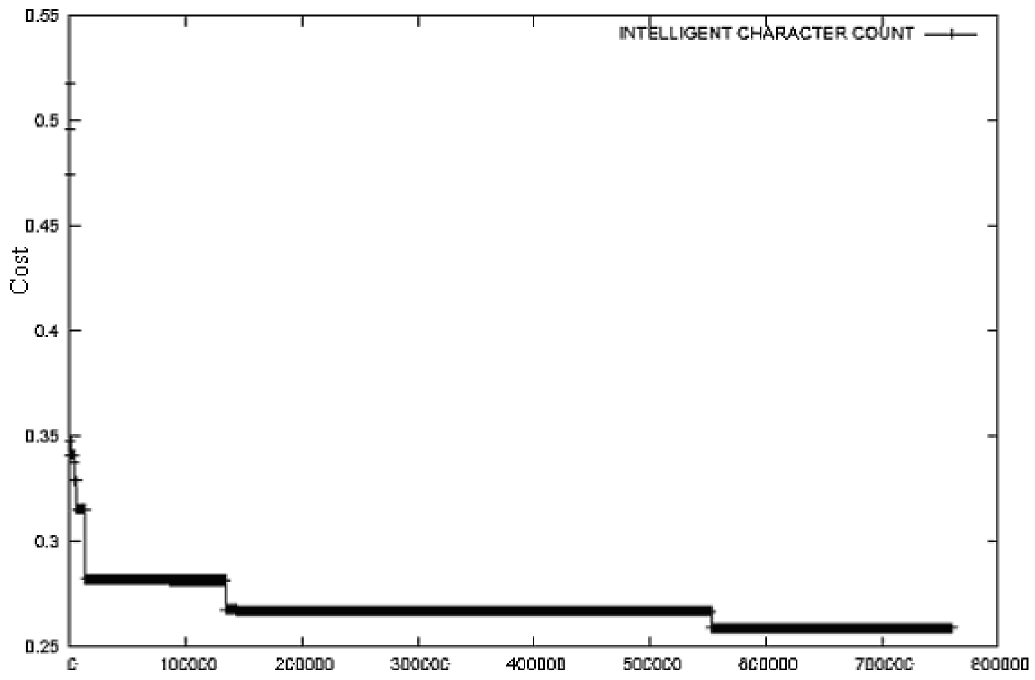

Figure 9: PSO with groups for cryptanalysis of round of DES without Sbox, population=30, pincrement=2, gincrement=4, length of text=3826 bytes, initial cost=0.518019, cost at saturation=0.25910

5.4 The Group PSO

Here we discuss a unique variation of Particle Swarm Optimization. In the PSO algorithm that we have seen, there is a single swarm made up of large number of particles. Experiments show that increasing the number of particles in a single swarm does not give better results. It only adds to more computation. In order to divide the search space, we explore the option of "group of swarms", a concept proposed by us for the first time.

In this concept, there are many swarms active at a given time. The swarms are initialized in such a way that they are evenly distributed across the search space. In ideal case, their movements should not overlap. To achieve this, proper distribution of swarms across the search space is important. Each of the swarms moves through the search space independently irrespective of other swarms. Each swarm has its own global best and rest of the particles in that swarm try to follow that global best.

The algorithm provided remains the same except that there are many versions of this algorithm running at the same time. This is an ideal situation for multiprocessor systems or grids. However, in our experiments we have implemented the group swarm algorithm for sequential execution. The speedup achieved by the parallel version or on grid could be an interesting thing to investigate. Since there is little communication between swarms, there are many opportunities to modify the algorithm so that it suits well for parallel or distributed execution.

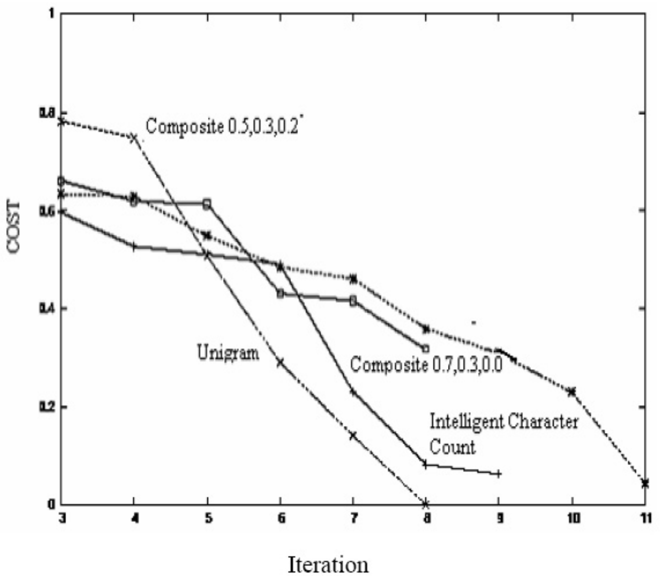

Figure 10: SDES cryptanalysis using SA, text length=3856 bytes, time taken=25 seconds for combined fitness function of unigram and intelligent character count

5.5 Experiments and Results

PSO and group PSO were used for cryptanalysis of 4-round DES without Sbox. We summarize the results of these experiments in this subsection. The experiment shown in Figure 11 was performed on 4-round DES without S-box. The fitness function used was unigram. As we can see, the fitness value has started from about 0.5 and come down to 0.3 in about 3250 generations. Also, we can see a rapid decrease in fitness function at the beginning and the rate of decrease decreases as we run the experiments for longer period. This is because as the fitness value decreases, it becomes more and more difficult to find a better key. This is the problem with all the algorithms. PSO, in spite of being so simple, has performed better than SA.

5.6 Experiments and Results

PSO and group PSO were used for cryptanalysis of 4-round DES without Sbox. We summarize the results of these experiments in this subsection. The experiment shown in Figure 11 was performed on 4-round DES without S-box. The fitness function used was unigram. As we can see, the fitness value has started from about 0.5 and come down to 0.3 in about 3250 generations. Also, we can see a rapid decrease in fitness function at the beginning and the rate of decrease decreases as we run the experiments for longer period. This is because as the fitness value decreases, it becomes more and more difficult to find a better key. This is the problem with all the algorithms. PSO, in spite of being so simple, has performed better than SA.

Also, PSO outperforms all other optimization techniques in discrete optimization problems. In this case also, we have observed similar findings. As we will see in the next experiment, group PSO performs slightly better than PSO.

Figure 12 shows the result of group PSO algorithm. In this case, search space was divided into 100 swarms. At the beginning, it was ensured that swarms start at different positions. But it is very difficult to ensure that trajectories of the swarms do not intersect during the execution. The X- axis here does not show the number of generations, but it shows the number of generations multiplied by the number of swarms, i.e. 100.

As we can see, the fitness value has started from above 0.55 and has come down to 0.22. Number of function evaluations is also quite high in this case. Also, the saturation is not prominent in Figure 12. The decrease in fitness value continues even after 3000 iterations.

Thus, PSO is a very promising approach for solving the problems of cryptanalysis of DES and any discrete optimization problem in general. However, this is a fairly new technique and there is lot of scope for research in this area. One of the ideas suggested in this paper, is the exploration of group PSO in a parallel execution environment.

Table 2: 16-bit DES encryption/decryption algorithm

```
INPUT:16-bit data block,16 bit key
1. (Key Schedule) compute four 12 bit keys.
2. $(L0, R0) = IP(m1, m2, \ldots, m16)$, using Table 3,
     $L0 = m10 \ldots m8$
     $R0 = m9 \ldots m7$.
3. For $i$ from 1 to 4, compute $Li$ and $Ri$ as shown
   below.
   a. Expand $ri - 1$ from 8 to 12 bits using $EP$ shown
      in Table 6.
      $T \leftarrow E(Ri - 1)$
   b. $T' \leftarrow T \oplus Ki$
   c. $T'' \leftarrow (S1(B1), S2(B2))$. Here $B1$ and $B2$ are
      first and last 6-bits of $T''$(for SBOX refer Table
      8).
      Using SBOX is identical to DES.
d. $T''' \leftarrow P(T'')$ using Table 7.
4. $b1b2...b16 \leftarrow (R4, L4)$ (Exchange final blocks $L16$,
   $R16$)
5. $C \leftarrow IP^{-1}$ $(b1b2 \ldots b16)$using Table 3.
```

The group PSO as compared to SA with thermostatistical persistency performs better in terms of percentage of successful attacks, overall execution time, and need to tune various parameters of the algorithm leading to simple program development effort.

6 Cryptanalysis of Simplified DES

Since attacking DES with the heuristic techniques seems to be harder due to the compute-intensive nature of the problem, we have implemented SDES [6] and used heuristics for cryptanalysis. The results of SDES cryptanalysis are shown in Figure 10. In the experiments conducted, we could retrieve the key and plaintext in all the cases. We tried a combined fitness function based on the unigram statistics and the number of ASCII representable intelligible character count.

6.1 16-bit Key DES

We have designed 16-bit DES algorithm and used heuristics for cryptanalysis of this cipher. The details are shown in Tables 3 to 8.

```
Algorithm: Key Schedule
INPUT: 16-bit key $K = K1 \ldots K16$
OUTPUT: Four 12-bit keys $Ki$, $1 \le i \le 4$
1. Define $vi$, $1 \le i \le 4$:$vi = 1$ for $i \le$ ; $vi = 2$ otherwise.
2. $T < -PC1(K)$; using Table 4 represent $T$ as 8-bit
   halves $(C0, D0)$
3. For $i$ from 1 to 4 compute $Ki$ as follows:
   $Ci \leftarrow (Ci - 1 \leftarrow vi)$
   $Di \leftarrow (Di - 1 \leftarrow vi)$
   $Ki \leftarrow PC2(Ci, Di)$
```

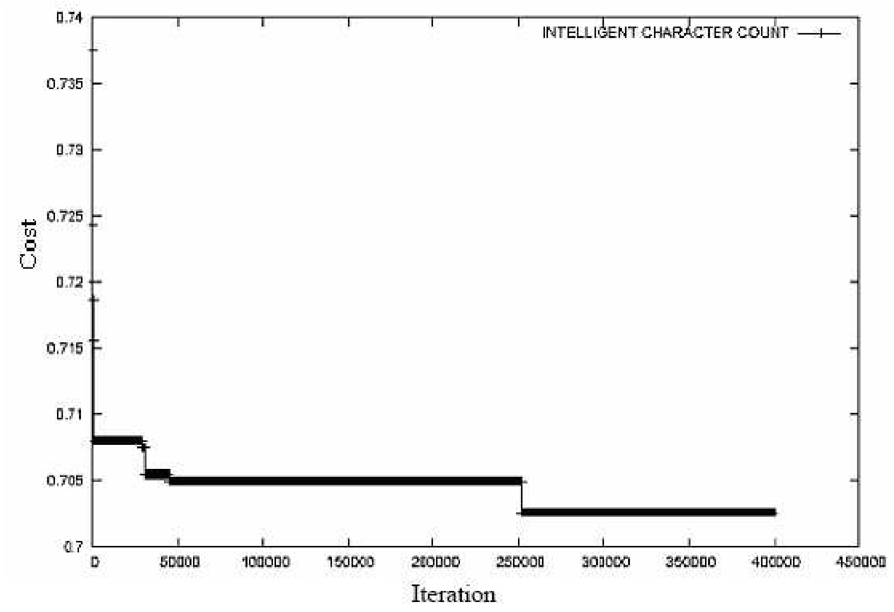

Figure 11: Convergence of PSO algorithm

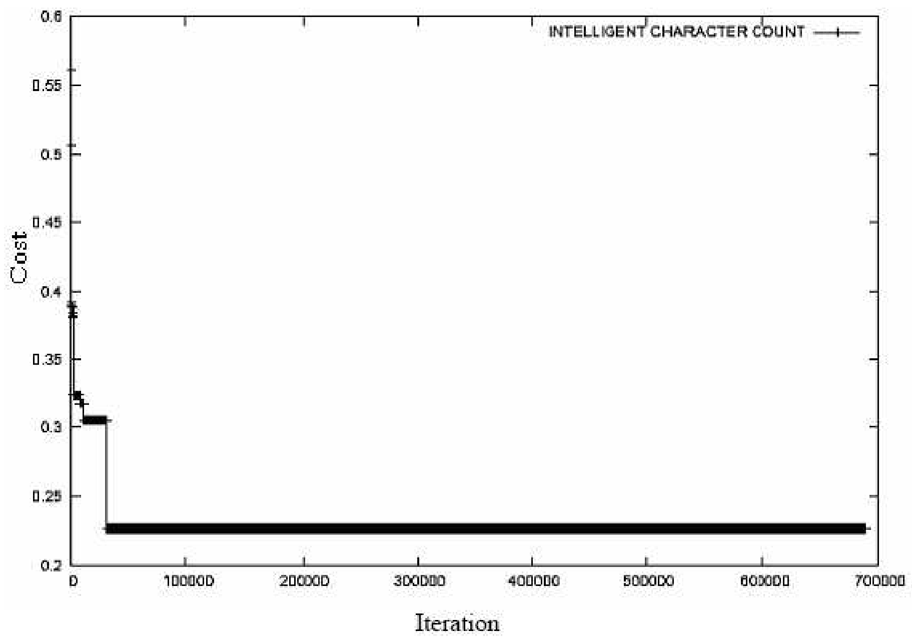

Figure 12: Performance of group PSO

Table 3: IP and IP^{-1}

10	2	12	4	14	6	16	8
9	1	11	3	13	5	15	7

Table 4: PC1

9	1	10	2	11	3	15	7
14	6	13	5	12	4	8	16

6.2 Cryptanalysis of 16-bit DES

16-bit DES is subjected to cryptanalysis using the SA technique, with variation of the fitness functions. The results are shown in Figures 13 to 18. In these experiments, the fitness function used is a combination of monogram and intelligible ASCII character set. We have embedded some concepts of thermostatistical persistency into these SA experiments.

7 Other Related Approaches

Matsui [5] has reported results on linear cryptanalysis for DES cipher and concludes that it is possible to attack DES cipher with 2^{43} known-plaintexts. Matsui [5] also reports that if the plaintexts consist of natural English sentences represented by ASCII codes, 8-round DES cipher is breakable with 2^{29} ciphertexts only. Biham and Shamir [1] gave a general method for chosen plaintext attacks—the differential cryptanalysis. Using a deep analysis of the internal framework of the function, they use a chosen bit-wise exclusion OR difference between two texts. Their main result proves that it is possible to mount an attack on DES with 2^{47} chosen plaintexts. Both methods are based on the concept of characteristic. This depends on the propagation of the correlated piece of information. It is associated to a probability which needs to be as biased as possible.

Vaudenay [8] describes a method of statistical cryptanalysis of DES, a combination and improvement of both linear and differential cryptanalysis and suggests that the linearity of Sboxes is not important. This study will motivate more research in the use of statistic experiments in cryptanalysis.

We also studied linear and differential cryptanalysis of a variant of SDES. To obtain a linear equation with high

Table 5: PC2

1	5	15	6
10	12	4	8
16	7	13	2

Table 6: Expansion permutation

1	2	3	4
3	4	5	6
5	6	7	8

Table 7: Permutation P8

7	1	5	2
8	3	6	4

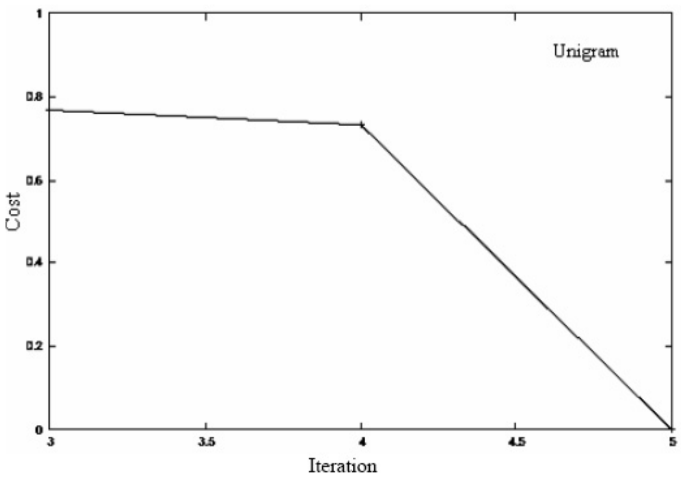

Figure 13: SA for cryptanalysis of 16 bit DES. Length of input text=3856 bytes, starting cost=0.713767, final cost=0.062484, initial temperature=20, final temperature=19.980021, time taken=57.033026 seconds, successful cryptanalysis.

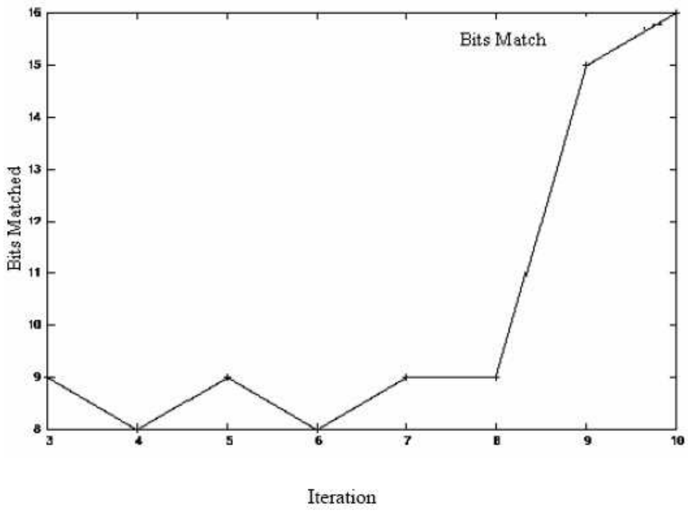

Figure 14: Number of bits matched during experiment

Table 8: SBOX

S_0															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_1															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

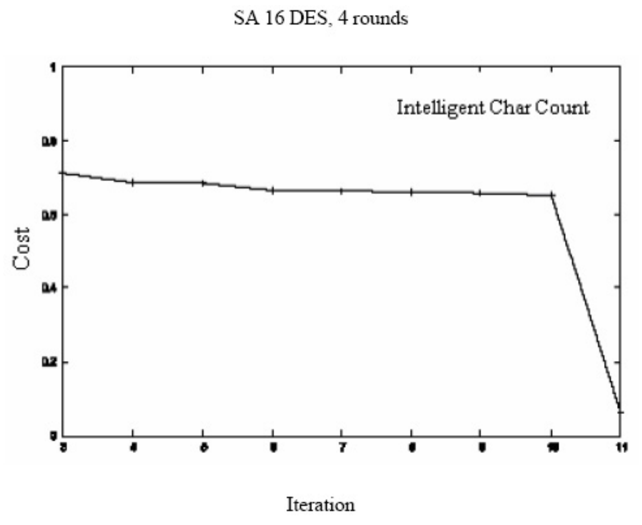

Figure 15: SA for cryptanalysis 16 bit DES, with fitness function as Unigram Length of input text=3856 bytes, time=508.060787 seconds. Successful cryptanalysis.

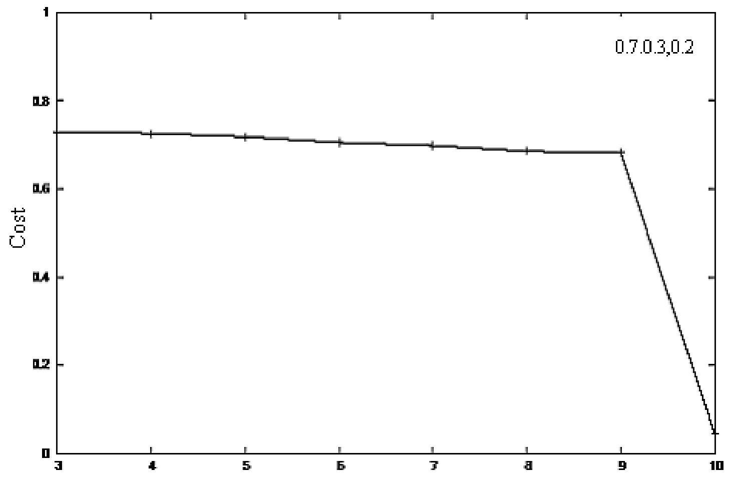

Figure 17: SA for cryptanalysis 16 bit DES, with composite fitness function. Length of input text=3856 bytes, time=240.025867 seconds. Successful cryptanalysis.

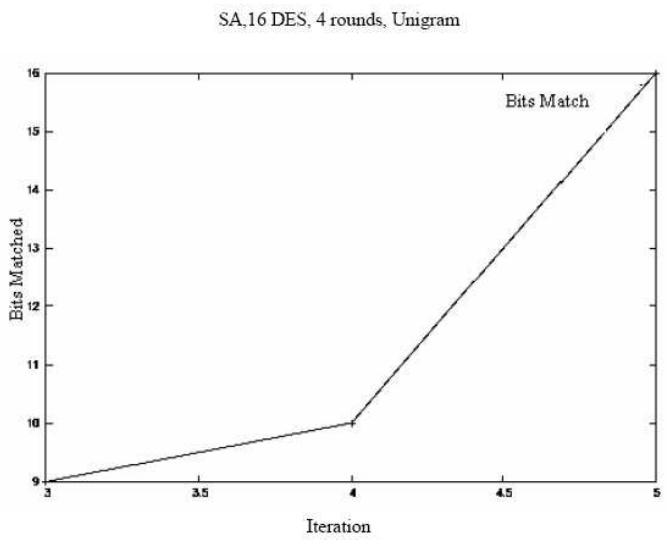

Figure 16: Number of bits matched.

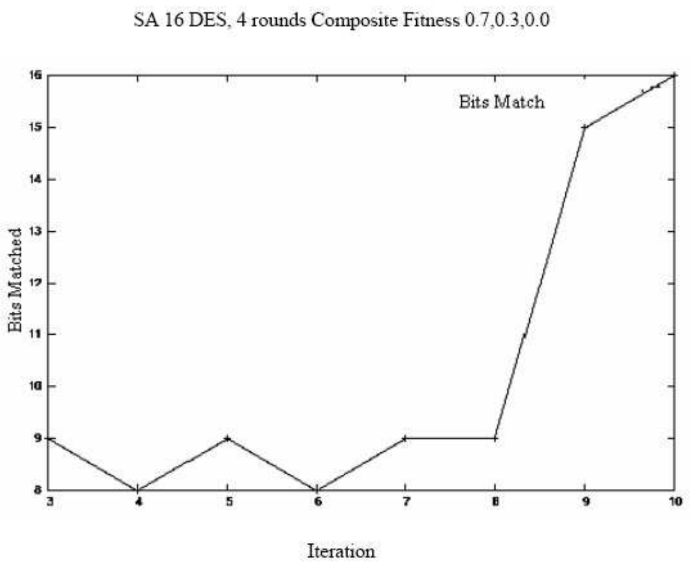

Figure 18: Number of bits matched.

probability bias, we start with a statistical linear path between the input and output bits of each Sbox. We then extend this path to the entire cipher and finally reach a linear expression without any intermediate value. For the differential cryptanalysis, the attack was successful and we were able to extract the entire subkey of round 2. These subkey bits are the actual 8 bits of the SDES 10 key bits, 2 bits still missing. The linear cryptanalysis too was successful. Increase of plaintexts increases the success rate of the method.

Both linear and differential cryptanalysis methods need handling of large number of plaintexts.

8 Conclusions

Cryptanalysis has been formulated by several researchers as an interesting optimization problem. Due to its complex nature, several heuristics can be used to solve this problem. However, it is hard to quickly demonstrate success of these methods when applied to some of the complex ciphers such as AES. At the same time, there is a need to develop such techniques as a viable alternative to brute force method, linear and differential cryptanalysis. Thus there is a case to systematically develop these techniques and use them to progressively simple to complex ciphers. Towards this objective, we have demonstrated that optimization heuristics such as simulated annealing, its extension such as thermostatistical persistency and a recent method such as particle swarm, hold promise in cryptanalysis studies. We intend applying these methods to more complex ciphers.

References

[1] E. Biham and A Shamir, "Differential cryptanalysis of the full 16-round DES", in *Crypto'92*, LNCS 740, pp. 487-496, Springer-Verlag, 1993.

[2] P. Chardaire, J. L. Lutton, and A. Sutter, "Thermostatistical persistency: A powerful improving concept for simulated annealing algorithms", *European Journal of Operations Research*, vol. 86, pp. 565-579, 1985.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization", in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995.

[4] S. C. Krirkpatrick, J. D. Gellatt, and M. P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[5] M. Matsui, "The first experimental cryptanalysis of the data encryption standard", in *Crypto'94*, LNCS 839, pp. 1-11, 1984.

[6] E. Schaefer, "A simplified data encryption standard algorithm", *Cryptologia*, vol. 20, no. 1, pp.77-84, 1996.

[7] B. Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons, 1996.

[8] S. Vaudenay, "An experiment on DES statistical crypatanlysis", *Ecole Normal Superieure*, Technical Report, France, 1996.

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 currently pursuing her Ph.D with the Visvesvaraya Technological University, Belgaum, India. Also she is working as an Assistant Professor in the Department of Computer Science and Engineering, 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 Optimization Heuristics.

G. Raghavendra Rao Completed his BE, ME and Ph.D from University of Mysore, Indian Institute of Science, Bangalore and 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 and Engg. at the 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.