Cryptanalysis of Pointcheval's Identification Scheme Using Ant Colony Optimization

Mohammad Faisal Uddin, *Student Member, IEEE* and Amr M. Youssef, *Senior Member, IEEE*

***Abstract*— We investigate the use of a binary version of ant colony optimization for the cryptanalysis of an identification scheme based on the permuted perceptron problem (PPP) proposed by Pointcheval. Based on our experimental results, ACO-based attacks proved to be very effective on recovering the secret key of this scheme for various sets of PPP parameters.**

I. Introduction

A variety of heuristics optimization techniques (such as simulated annealing, tabu search, and genetic algorithms) have been widely used in the field of automated cryptanalysis of classical ciphers and for the construction of cryptographic Boolean functions (e.g., [1], [2], [3]).

More recently, the application of these techniques have been extended to the cryptanalysis of relatively stronger modern ciphers (e.g., [4], [5]).

In this paper, we investigate the use of ant colony optimization (ACO) [6] for the cryptanalysis of Pointcheval identification scheme [7]. In particular, we modify the traditional ACO algorithm to fit the binary nature of the problem under consideration. Based on our experimental results, ACO-based attacks proved to be very effective on recovering the secret key of this scheme for various sets of suggested PPP parameters.

The rest of the paper is organized as follows. In the next section, we briefly review the Permuted Perceptron Problem (PPP) introduced in [7]. In section III, we introduce the modified binary version of ACO. Some related works are reviewed in section IV. Our proposed search algorithm and experimental results are given in section V. Finally, our conclusions are given in section VI.

II. Pointcheval Identification Schemes

A cryptographic identification scheme [8] is a protocol that enables one party (e.g., Alice) to prove her identity interactively to another party, i.e., proves that she is really Alice. To prove her identity, Alice proves that she knows a secret key associated to her public key.

Several zero-knowledge based identification schemes have been proposed [9]. Theses schemes are based on number theoretic ideas and are typically not very attractive in terms of their computational requirements. On the other hand, other schemes based on NP-complete problems (such as Permuted Kernels Problem, Syndrome Decoding, and Constrained Linear Equations) seem to be are more efficient from the computational point of view. Pointcheval [7] introduced an identification scheme based on the Perceptrons Problem, an NP-complete problem, which seems to be well suited for resource constrained devices such as smart cards. In the following subsection, we briefly review the definitions and notations in [7] that are relevant to our attack.

A. The Permuted Perceptron Problem

A vector whose entries have value either +1 or -1 is called ϵ-vector, and similarly for matrices. If X is a vector of size m, let X_i denote the i^{th} entry in X.

Definition 1: The Perceptron Problem (PP):

Given an $m \times n$ ϵ -matrix, A, find an ϵ-vector V of size n such that $(AV)_i \geq 0$, for all $i = 1, \cdots m$.

Definition 2: The Permuted Perceptron Problem (PPP):

Given an $m \times n$ ϵ-matrix, A, and a multiset S of nonnegative numbers of size m, find ϵ-vector V of size n, such that $\{\{(AV)_i | i = \{1, \cdots, m\}\}\} = S\}$.
Obviously, a solution for PPP is a solution for PP. This lead to the conclusion that the Permuted Perceptron Problem is more difficult to solve than the original Perceptron Problem. However, as pointed by several researchers, knowledge of the prescribed multiset S may give some hint to a solution of PPP.

For cryptographic applications, one needs instances for which a solution is known. To get such instances, a (pseudo-) random ϵ-vector V of size n is chosen which will be a solution of the future instance. Hereafter a

M. Faisal Uddin is with the Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada (email: mf_uddinr@encs.concordia.ca).

Amr Youssef is with the Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Canada (email: youssef@ciise.concordia.ca).

1-4244-1340-0/07/$25.00

random ϵ-matrix A of size $m \times n$ is generated and modified in the following way:

For $i = 1, \cdots m$

- If $(AV)_i < 0$, the i^{th} row of A is multiplied by -1.
- If $(AV)_i \geq 0$, the i^{th} row of A remains unchanged.

Finally the multiset $S = \{\{(AV)_i | \{i = 1, \cdots m\}\}\}$ is computed. Consequently, (A, S) is an instance of the Permuted Perceptron Problem, with V as a solution.

In an identification protocol, each prover uses a public instance (A, S), and a secret key V. To convince a verifier of his identity, a prover gives him evidence of a solution V to the instance (A, S) by using a zero-knowledge protocol. The description of several such protocols based on PPP is detailed in [7].

The smallest size recommended for the matrix A is $m = 101$, $n = 117$. Based on the analysis presented in [7], Pointcheval concluded that the complexity of solving instances of this size is about 2^{64} elementary operations, hence a work load sufficient to guarantee the security of the underlying protocol.

III. Ant Colony Optimization

Ant Colony Optimization [6] is a heuristic optimization method for solving different combinatorial optimization problems with a permutation search space. It is a population based approach which borrows ideas from biological ants. The social behaviors of ants have been much studied by the scientists and from their behavior the computer scientists came up with the idea of this optimization. Experiments with real ants showed that ants go from the nest to the food source and backwards, then after a while, the ants prefer the shortest path from the nest to the food source. Real ants are capable of finding the shortest path from their nest to a food source without using visual cue. Ants have a special way of communicating information concerning food sources. While walking, ants secrete an aromatic essence on the ground, called pheromone. The other ants will follow the path of greater pheromone trail with higher probability and as they will follow the path they as well will secrete pheromone there. So the pheromone of that path which greater number of ants are following will increase and as the pheromone trail of that path increases more ants will follow that path. Since ants passing through food source by shorter path will come back to the nest sooner than ants passing through longer paths, the shorter path will have a higher traffic density than that of the longer one. Thus a single ant will follow the shortest path with higher probability. Dorigo and Gambardella [10] presented ant colony system (ACS) for solving travelling salesman problem (TSP).

In this paper, we introduce a simple variant of the ACO algorithm that allow us to apply it to optimization problems with a binary search space. The system is initialized with a group of ants moving across a full binary tree of depth n and 2^n leaves. Each leaf corresponds to one of the possible solution. The root of the tree represents the nest of the ants and the cost of the solution associated with each leaf corresponds to the amount of food found at the food source. The algorithm proceeds by iterating through the following three basic steps:

- Construct a solution for all ants: At each node, each ant has to make a (statistical) decision whether to follow the right path or the left path. At the first iteration, all the ants will move randomly. However, on subsequent iterations, the ants' choices will be influenced by the intensity of the pheromone trails left by preceding ants. A higher level of pheromone on the right path gives an ant a stronger stimulus and thus a higher probability to turn right and vice versa. Let $Pher_l(R)$ and $Pher_l(L)$ denote the value of the pheromone accumulated at the right edge and the left edge of a given node at the l^{th} level of the tree. Then the ants' behavior equivalent to having each ant choosing a uniformly distributed random variable $0 \leq r \leq 1$ and choosing to follow the right edge at the l^{th} level of the tree if
$$r \geq \frac{Pher_l(R)}{Pher_l(R) + Pher_l(L)} \quad (1)$$
and to follow the left edge otherwise.
- Do a global pheromone update: This step is slightly different than the one proposed in the original ANT colony optimization algorithm. Instead of updating the pheromone along the visited arcs only, we update all the corresponding 2^{l-1} arcs at the l^{th} level of the tree.
- Evaporate pheromone: After each iteration, a portion of the pheromone of the edge is evaporated according to a local updating rule, such that the probability of the selection of that edge by other ants decreases. This prevents construction of similar paths by the set of ants and increases the diversity of the system. The rate of evaporation provides a compromise between the rate of convergence and reliability of convergence. Fast evaporation causes the search algorithm to be stuck at local optima, while slow evaporation lowers the rate of

convergence. After enough number of iterations (call it itt) of the algorithm, the pheromone of the good edges which are used in constructing of solutions corresponding to better cost functions will increase and the pheromone of the other edges will evaporate. Thus, in the higher iterations, the probability of constructing better solutions increases.

IV. Related Work

Knudsen and Meier [11] described an attack on Pointcheval's scheme which can recover the secret key in time much faster than estimated by its designer. The basic idea in [11] is to use several applications of a simulated annealing algorithm and combine the outcomes into an improved search.

Clark and Jacob [12] showed how fault injection (i.e., the use of cost functions that define problems slightly different to the target one as a means of gaining information on the target problem's solution) and timing analysis can be interpreted for a simulated annealing attack on Pointchevals Permuted Perceptron Problem (PPP) identification schemes. All recommended sizes of the PPP schemes were shown to be unsafe.

In this work, we borrow several ideas from the above works. In [11], the following cost functions were used to solve the PP-problem and the PPP-problem respectively.

$$C_{PP}(V) = g \times \sum_{i=1}^{m} |(AV')_i| - (AV')_i) \qquad (2)$$

$$\begin{aligned} C_{PPP}(V) &= g_1 \times \textstyle\sum_{i=1}^{m} |(AV')_i| - (AV')_i) \\ &+ \quad g_2 \times \textstyle\sum_{i=1}^{n} (|H_i - H'_i|) \end{aligned} \qquad (3)$$

Typically, $g1$ is much greater than $g2$ (e.g., (g_1, g_2) can be set to $(30, 1)$). The second part of the PPP cost function above compensates for the distance between the correct and candidate histogram vectors. It should be noted that the histogram part of the cost function, $\sum_{i=1}^{n}(|H_i - H'_i|)$, can change too much even if only a single sign in V is changed. Thus the cost function is not as continuous for ACO to work in a straight forward way (e.g., by using multiple restarts). On the other hand, for the PPP problem, we can not rely only on the first part of the cost function because, for several selections of the algorithm parameters (n, m), this part of the cost function vanishes for many vectors in the search space which satisfy the PP problem without satisfying the PPP problem. In fact, it is this inherent nature of the PPP problem that makes it hard to find a solution because most search algorithms will tend to get stuck at local optima corresponding to PP solutions that are relatively distant from the target PPP solution.

V. The search algorithms

A. PP Problem

For Perceptron Problem (PP), the basic ACO search described in the previous section finds the solutions very easily. The experiment is done for problems of following dimensions $(m, n) = \{(101, 117), (121, 137), (151, 167)\}$. For each dimension 50 randomly selected problem sets are generated and binary version of ACO finds a solution every time in few seconds. It is worth noting that the solution found may not be identical to the one initially generated as the problem instance.

B. PPP Problem

Applying the basic ACO search does not seem to be effective in solving the PPP problem. Even with a large number of multiple restarts, the ants tend to get stuck at local optima (satisfying the PP constraint or close to it) but far away from the target PPP solution. In order to avoid this, we divided our search into the following three phases.

- In the first phase, we run the ACO with a relatively small number of ants t times (this t should not be confused with the internal number of iterations, itt, of the ACO algorithm). Each time we start with a different initial ANT population and a slight variation in the cost function weight parameters g_1, g_2. For each one of these t runs, we record the candidate vector with the lowest cost function. Then we find the consistent positions (i.e., the set of positions that have the same sign) among the t candidate vectors.
  The main reason for perturbing the cost function for each iteration is based on our experimental observation that, for a given cost function, some bits in the solution have a tendency to get stuck at a wrong value throughout the search process irrespective of the initial starting positions of the ants. By perturbing the cost function, we aim to prevent this sort of consistent errors.
  Table I shows the average number of consistent positions obtained for different values of (m, n), 50 random instances each. It also shows the minimum and maximum values of the number of consistent positions.
- In the second phase, we run the ACO again (this time with a larger number of ANTS) but, instead

TABLE I

AVERAGE NUMBER OF CONSISTENT POSITIONS AND RANGE (EXPERIMENTAL SET I)

(n, m)	Average	(min,max)
$(121, 81)$	72.2	(56,81)
$(101, 81)$	60.95	(49,81)
$(73, 73)$	47.1	(40,73)
$(81, 81)$	53.09	(46,81)
$(101, 101)$	65.7	$(55, 73)$
$(101, 117)$	72.67	$(64, 79)$

of starting with a random initial population, the consistent positions obtained from phase one are initialized with the corresponding solution and the other positions are initialized randomly.

- The third phase is executed only if the solution was not found in the second phase. In this case, we flip b consistent positions ($b = 1, \cdots,$ total number of consistent positions) and run the ACO with the same parameters as in phase 2 for 3 more times. If a solution was not found by the end of phase 3, we return a failure. Based on our experimental observations, if a solution is to be found by the ants in this stage, it was found very quickly. This observation allowed us to terminate the search quickly if no solution was likely to be found efficiently (hence our success ratio is some what conservative).

Note that phase 2 is actually a special case of phase 3 (with $b = 0$). This distinction is just made for clarity purpose.

Table II shows the success ratio as well as the estimated expected value for the number of ACO runs during the third phase (for the successful cases) calculated from the distribution of the number of errors in the consistent bits. For example, if the number of errors in the consistent bits is N_e, then in, the worst case, a successful termination of phase 3 would require $\sum_{i=0}^{N_e} \binom{n}{i}$ ACO runs.

TABLE II

SUCCESS RATIO AND EXPECTED NUMBER OF ACO RUNS IN PHASE 3 (EXPERIMENTAL SET I)

(n, m)	Success ratio	E(number of ACO runs)
$(121, 81)$	49/50	$2^{4.79}$
$(101, 81)$	44/50	$2^{9.88}$
$(73, 73)$	39/50	$2^{14.03}$
$(81, 81)$	35/50	$2^{16.01}$
$(101, 101)$	23/50	$2^{24.57}$
$(101, 117)$	12/50	$2^{33.27}$

Table III shows the distribution of the number of error bits (N_e) in the consistent positions for the cases in which phase 3 was not terminated with a failure. To explain the table, as an example for $(n, m) = (121, 81)$, in 28 times out of the 50 times, all the consistent bits were correct and in 11 times, we needed to correct one bit.

TABLE III

DISTRIBUTION OF THE NUMBER OF ERROR BITS (N_e) IN THE CONSISTENT POSITIONS (EXPERIMENTAL SET I)

N_e	(121,81)	(101,81)	(73,73)	(81,81)	(101,101)	(101,117)
0	28	9	3	2	0	0
1	11	10	4	2	0	0
2	3	10	7	6	2	0
3	3	8	11	7	1	1
4	2	5	4	11	2	1
5	1	0	5	0	8	0
6	1	1	3	4	2	3
7	0	0	1	2	5	0
8	0	1	0	1	2	1
9	0	0	1	0	1	2
10	0	0	0	0	0	2
11	0	0	0	0	0	0
12	0	0	0	0	0	1
13	0	0	0	0	0	0
14	0	0	0	0	0	0
15	0	0	0	0	0	0
16	0	0	0	0	0	1

In all the above set of experiments (set II), t was set to 20, the number of ants was set to 50 in phase 1 and to about 3000 in phase 2 and 3. In another set of experiments, we slightly modified phase 1 of the search algorithm. Our modification is based on the following observation [7] [11]:

Let $A = (a_{ij})$ be an $m \times n$ ϵ-matrix. Let $s_j = \sum_{i=1}^{m} a_{ij}$ for $j = 1 \cdots n$. Let $M_j = sign(s_j)$, $j = 1 \cdots M$.

Suppose (A, S) is an instance for PPP with solution V. Then the vectors M and V are correlated [7]:

$$\#\{j | M_j = V_j, j = 1 \cdots n\} \approx 0.8n.$$

By inspection of the majority vector and the solution vector it was noted that [11] this fraction increases for larger values of $abs(s_j)$. As an example, for instances of PPP where $m = n = 73$, experiments show that

$$Prob(M_j = V_j | abs(s_j) \geq 11) \approx 0.94$$

whereas for $m = n = 101$ this probability is still 0.92. Moreover for $m = n = 73$ and $m = n = 101$ tests show that on the average $abs(sj) \geq 11$ for 28 respectively 46 entries, or 38% respectively 46% of the entries. Similarly for $m = 101$, $n = 117$, this probability is about 0.91 and that on the average $abs(s_j) \geq 11$ for about 44% of the entries [11].

Since the vector S can be computed from the public matrix A, thus in this case, instead of starting with a random initial population, we initialize a large percentage (about 95% of the initial population in our experiments) with this majority vector in all positions $1 \leq j \leq n, abs(s_j) \geq 11$. The other positions are initialized at random as well as the rest of the ants. Then phase 2 and 3 proceed exactly as before.

Tables IV, V and VI show the corresponding results for this experiment. It is clear that this modification allows us to improve the success ratio dramatically for three parameters in table 1 at the expense of a larger search space in phase 3. We only report the results corresponding to the last four selections of (n, m) because for $(m, n) = (121, 81), (101, 81)$, this modification worsen both the success ratio and the search space.

TABLE IV
AVERAGE NUMBER OF CONSISTENT POSITIONS AND RANGE (EXPERIMENTAL SET II)

(n, m)	Average	(min,max)
$(73, 73)$	56.24	(50,67)
$(81, 81)$	62.86	(51,81)
$(101, 101)$	80.83	$(71, 91)$
$(101, 117)$	90.20	$(79, 99)$

TABLE V
SUCCESS RATIO AND EXPECTED NUMBER OF ACO RUNS IN PHASE 3 (EXPERIMENTAL SET II)

(n, m)	Success ratio	E(number of ACO runs)
$(73, 73)$	50/50	$2^{22.87}$
$(81, 81)$	50/50	$2^{26.95}$
$(101, 101)$	48/50	$2^{36.91}$
$(101, 117)$	44/50	$2^{48.63}$

In all the above set of experiments, t was set to 10, the number of ants was set to 200 in phase 1 and to about 3000 in phase 2 and 3.

Also for all our experiments (set I and set II), the pheromone evaporation rate was set to 0.95 in all the three phases. The number of internal iterations, itt, was upper bounded by 2000 but usually the solution in a much smaller number of iterations.

It is worth noting that, in the case where $m < n$, our proposed algorithm is less successful. As explained in [11], this may be due to the fact that, for a given matrix A and a multiset S, there are several solutions to the problem and the search algorithm is not able to converge to one single solution.

TABLE VI
DISTRIBUTION OF THE NUMBER OF ERROR BITS (N_e) IN THE CONSISTENT POSITIONS (EXPERIMENTAL SET II)

N_e	(73,73)	(81,81)	(101,101)	(101,117)
1	1	0	0	0
2	2	6	0	0
3	7	0	1	0
4	11	5	2	0
5	8	7	4	0
6	6	9	7	0
7	3	6	5	3
8	8	4	4	3
9	1	7	6	3
10	2	2	4	2
11	0	2	6	6
12	0	0	1	7
13	0	2	3	5
14	0	0	3	6
15	1	0	0	0
16	0	0	1	4
17	0	0	0	1
18	0	0	0	1
19	0	0	1	1
20	0	0	0	2

VI. CONCLUSIONS

In this paper, we presented modified version of ACO suitable for solving binary optimization problems. To demonstrate the effectiveness of our algorithm, we applied it to the cryptanalysis of the identification schemes based on the permuted perceptron problem.

Although the insecurity of this scheme has been demonstrated previously, our main objective is to add one more tool (for further investigation) to the set of heuristic techniques already available for the cryptanalysis of modern ciphers. It is also interesting to apply the ideas presented in this attack to other similar schemes such as the one presented in [13].

On the other hand, one main disadvantage of heuristic optimization techniques (including ACO) is its large sensitivity to parameter variations. Although fine tuning of these parameters can be done by trial and error, it is interesting to find analytical formula for the optimal regions of these parameters.

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their valuable comments. This work is supported in part by the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] Andrew Clark, *Optimisation Heuristics for Cryptology*, PhD thesis, Queensland University of Technology, 1998.

[2] M.D. Russell, J.A. Clark, and S. Stepney, *Making the most of two heuristics: breaking transposition ciphers with ants*, The Congress on Evolutionary Computation (CEC '03), Vol. 4, pp. 2653- 2658, 2003.

[3] Z. Saber, M. Faisal Uddin and Amr Youssef, *On Some Resilient Functions Constructions using PSO-based Spectral Inversion*, Proc. of IEEE swarm intelligence symposium, special session of applications of swarm intelligence to number theory, pp. 38-42, 2006.

[4] E. C. Laskariab, G.C. Meletiouc, Y.C. Stamatioud and M.N. Vrahatisab, *Evolutionary Computation Based Cryptanalysis: A First Study*, Nonlinear Analysis: Theory, Methods and Applications, vol. 63, pp. e823-e830, 2005.

[5] E. C. Laskariab, G.C. Meletiouc and M.N. Vrahatisab, *Problems of Cryptography as Discrete Optimization Tasks", Nonlinear Analysis: Theory, Methods and Applications*, vol. 63, pp. e831-e837, 2005.

[6] M. Dorigo, V. Maniezzo, and A. Colorni, *The ant system: optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics-Part B 26(1) (1996), pp. 29-41.

[7] D. Pointcheval, *A new identification scheme based on the perceptrons problem*, In L.C. Guillou and J.-J. Quisquater, editors, Advances in Cryptology - EUROCRYPT'95, LNCS 921, pp. 319 - 328. Springer Verlag, 1995.

[8] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of applied cryptography*, CRC press, 2001.

[9] A. Fiat, A. Shamir, *How to prove yourself: practical solutions of identication and signature problems*, Advances in Cryptology - CRYPTO'86, LNCS 263, pp. 186-194. Springer-Verlag, 1987.

[10] M. Dorigo, L.M. Gambardella, *Ant colony system: a cooperative learning approach to the traveling salesman problem*, IEEE Transactions on Evolutionary Computation, vol. 1, no.1, pp.53-66, 1997.

[11] Lars R. Knudsen and Willi Meier, Cryptanalysis of an Identification Scheme Based on the Permuted Perceptron Problem, In J. Stern editor, EUROCRYPT'99, LNCS 1592, pp. 363-374, 1999. Springer-Verlag, 1999.

[12] John A. Clark and Jeremy L. Jacob, *Fault Injection and a Timing Channel on an Analysis Technique*, In L. Knudsen editor, Eurocrypt 2002, LNCS 2332 . pp. 181-196, 2002. Springer-Verlag, 2002.

[13] Shahrokh Saeednia, *New NP-Complete Partition Problems*, IEEE Trans. on Information Theory, Vol. 48, No. 7, pp. 2092-2094, July 2002.