2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Cryptanalysis of Simple Substitution Ciphers Using Particle Swarm Optimization

Mohammad Faisal Uddin and Amr M. Youssef

Abstract— **We investigate the use of Particle Swarm Optimization (PSO) in automated cryptanalysis of classical simple substitution ciphers. Based on our experimental results, PSO-based attacks proved to be very effective on various sets of encoding keys.**

I. INTRODUCTION

CLASSICAL ciphers [1],[2] are the earliest schemes of cryptography. These ciphers were developed and used before the computer age, and are usually carried out by pen and paper or by simple mechanical devices rather than by electronic media. There are usually two types of classical ciphers, namely transposition ciphers and substitution ciphers. Although, from the security point of view, classical ciphers are no match to the recently developed ciphers, they have not lost their importance because most of the commonly used modern ciphers use classical ciphers as their building blocks. In fact, several complex algorithms can be formed by mixing substitution and transposition operations. Modern block ciphers such as DES and AES iterate through several stages of substitution and transposition.

Particle swarm optimization (PSO) [3] is a population based, self-adaptive search optimization technique inspired by social behavior of bird flocking or fish schooling.

In the paper we investigate the use of PSO in automated cryptanalysis of simple substitution ciphers. The rest of the paper is organized as follows. In section II, we briefly review some of the previous work related to cryptanalysis of classical ciphers. In section III, we outline the simple substitution cipher. Section IV summarizes the n-gram statistics and the cost function used in our attack. In Section V, we review the PSO algorithm and describe how it is adopted for the cryptanalysis of substitution ciphers. In section VI, we explain our experimental setup and report the obtained results. Finally, section VII is the conclusion.

II. PREVIOUS WORKS

Cryptanalysis, from the Greek kryptós, "hidden", and analýein, "to loosen", is the art and science of breaking, i.e., decoding ciphertext into its corresponding plaintext, without prior knowledge of the secret key.

The Arabs were among the first to make significant advances in cryptanalysis. Early in the 15th century, an Arabic author, Qalqashandi, wrote down a technique for solving ciphers using the average frequency of each letter of the language [2].

Over the last twenty-five years, several optimization heuristics have shown promise for automated cryptanalysis of classical ciphers [4].

One of the early proposals was given by Peleg and Rosenfeld [5]. They modeld the problem of breaking substitution ciphers as a probabilistic labeling problem. Every coded alphabet was assigned probabilities of representing plaintext alphabets and they updated the probabilities using the joint letters. Using this scheme in an iterative way they were able to break the cipher.

Carrol and Martin [6] developed an expert system approach to solve simple substitution ciphers using hand-coded heuristics.

Forsyth and Safavi-Naini [7] recast the problem as a combinatorial optimization problem and presented an attack on simple substitution cipher using simulated annealing algorithm.

Spillman *et. al* [8] presented an attack on simple substitution cipher using genetic algorithm. Clerk [4] re-implemented the genetic algorithm and simulated annealing attack in order to compare them and also evaluate a third technique using *tabu* search.

Bahler and King [9] used trigram statistics and relaxation scheme to iterate towards the most probable key as previously done by Peleg and Rosenfeld.

Lucks [10] used a word pattern dictionary and search over it with the constraint that all ciphertext characters must decrypt to the same plaintext character.

Hart [11] improved upon this method by directing this combinatorial search towards more frequent English words.

Jakobsen [12] introduced a fast algorithm for the cryptanalysis of simple substitution ciphers based on a process where an initial key guess is refined through a number of iterations. In each step the plaintext corresponding to the current key was evaluated and the result was used as a measure of how close we are in having

Mohammad Faisal Uddin is with the department of Electrical and Computer Engineering, Concordia University, Montreal, Canada. (e-mail: mf_uddinr@encs.concordia.ca).

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

0-7803-9487-9/06/$20.00/©2006 IEEE

discovered the correct key.

Recently, Ant Colony Optimization (ACO) was used successfully in breaking transposition ciphers [13] and simple substitution ciphers [14].

III. SIMPLE SUBSTITUTION CIPHER

Simple substitution cipher is a well-known cryptosystem. It is the simplest form of substitution ciphers. Each symbol in the plaintext maps to a different symbol in the ciphertext [1]. The simple substitution cipher used in this work operates on the English alphabet of 26 letters ("A-Z").

We assume that all the punctuations and structure (sentences/paragraphs marks, space characters, and newline characters) are removed from the plaintext in order to hide these obvious statistics from the ciphertext.

TABLE 1: EXAMPLE OF A SIMPLE SUBSTITUTION CIPHER

Key: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z X N Y A H P O G Z Q W B T S F L R C V M U E K J D I
Encryption: Plaintext: PARTICLESWARMOPTIMIZATIONISAPOWERFULTOOL Ciphertext: LXCMZYBHVKXCTFLMZTZIXMZFSZVXLFKHCPUBMFFB

Let $\boldsymbol{x}$ be an n-character alphabet $\{x_0,x_1,x_2,x_3,\ldots,x_{n-1}\}$, and $\boldsymbol{y}$ is also an n-character alphabet $\{K(x_0), K(x_1), K(x_2), K(x_3),\ldots, K(x_{n-1})\}$, where $K: \boldsymbol{x}\rightarrow\boldsymbol{y}$ is a one to one mapping of every alphabet of x to the corresponding alphabets of y. Here 'K' is the cipher key function which can be looked at as a permutation of the 26 character. The transmitter enciphers the plaintext into ciphertext with a predetermined key function (K) and sends it to the receiver. The receiver deciphers the ciphertext to plaintext with the inverse key function (K^{-1}).

An example for a simple substitution cipher key and encryption operation is shown in Table 1.

IV. N-GRAM STATISTICS AND COST FUNCTION

There exist $26! \approx 4.03291461 \times 10^{26} \approx 2^{88}$ possible keys for a simple substitution cipher with alphabet size of 26 characters. This number is obviously too large to allow any kind of exhaustive search.

However, one special property of simple substitution cipher that makes it relatively easy to cryptanalyze, is that the language statistics remain unchanged by the encryption process and hence frequency analysis presents a basic tool for breaking classical ciphers. In natural languages, certain letters of the alphabet appear more frequently than others. The n-gram statistics indicates the frequency distribution of all possible instances of n adjacent characters [3]. For example 'E' is the most common unigram (1-gram) in English language, and one of the common bigram (2-gram) in English language is 'TH'. These n-gram statistics can be used to measure the fitness of a suggested decryption key.

In our case we have only considered unigram and bigram statistics for deciphering the ciphertext. These statistics can be easily calculated from any large English corpus.

For our experiments, these statistics were generated from an online version of the book "Twenty Thousand Leagues Under the Sea" by Jules Verne.

Let R^U, R^B and DK^U, DK^B be the reference language unigram and bigram statistics and decrypted message unigram and bigram statistics (using a key K) respectively. Then, our cryptanalysis problem corresponds to finding a decryption key K such that minimizes the following weighted objective function

$$Cost(K) = \lambda_1 \sum_{c\in\{A,B,\ldots,Z\}} \left| R^U_{(c)} - DK^U_{(c)} \right| + \lambda_2 \sum_{c_1,c_2\in\{A,B,\ldots,Z\}} \left| R^B_{(c_1,c_2)} - DK^B_{(c_1,c_2)} \right|$$

V. PARTICLE SWARM OPTIMIZATION (PSO)

Particle swarm optimization [2] is a population based, self-adaptive search optimization technique inspired by social behavior of bird flocking or fish schooling.

Like other population-based optimization methods such as genetic algorithm, the particle swarm algorithm is initialized with a population of random solutions in the search space and searches for optimum solution by adjusting potential solutions over generations.

In PSO every potential solution are called particles. Each particle keeps track of its coordinates in the search space which are associated with the best solution it has achieved so far. This value is called particle best or *pbest* . Another value is the best value achieved so far by any particle in the population. This value is called global best or *gbest* .

After finding the two best values each particle updates its velocity ($v_{i,j}$) and position ($P_{i,j}$) towards its *pbest* and *gbest* locations as follows:

Particle velocity update:

$$v_{i,j} = c_0 v_{i,j} + c_1 r_1 (Ppbest_{i,j} - p_{i,j}) + c_2 r_2 (Pgbest_{i,j} - P_{i,j})$$

Particle position update:

$$P_{i,j} = P_{i,j} + v_{i,j}$$

Where, $Ppbest_{i,j}$ and $Pgbest_{i,j}$ are the particle best and global best position of the particles respectively. $1 \le r_1, r_2 \le 0$ are uniformly distributed random variables and c_0 , c_1 , c_2 are learning factors. These learning factors made PSO an attractive optimization technique. By varying these factors, it is possible to use PSO in a wide variety of applications.

In traditional PSO, the particle is encoded as a string of positions, which represent a multidimensional space. All the

dimensions are typically independent of each other, thus the updates of the velocity and the particle are performed independently in each dimension. However, in our cryptanalysis problem each solution in the parameter space represented by each particle is a permutation of the alphabetic characters corresponding to a key and hence the elements are not independent of each other.

Different particle update strategies were proposed in order to adapt PSO to permutation problems.

In [15],[16], it was suggested that a simple mapping from search space to permutation space can be constructed by sorting the elements in the particle position vector to get an ordered set $(P_{i\pi_1}, P_{i\pi_2}, \ldots P_{i\pi_n})$ where $P_{i\pi_1} \geq P_{i\pi_2} \geq \ldots P_{i\pi_n}$ and hence the permutation $(\pi_1, \pi_2, \cdots, \pi_n)$.

When trying the above approach, our experiments indicate that the non linearity inherent in the sorting process limits the success of the approach suggested in [15], [16] when the size of the permutation grows above 20 elements.

Another strategy was proposed by Hu *et al* [17]. In this work, we adopt a slightly modified version of this strategy.

In typical PSO systems, for updating the position of the particle, the velocity is added to the particle on each dimension. When the velocity is larger the particle may explore distant areas. In the new updating method [17], it also does the same thing in a bit different way. When the velocity is larger, the particle is more likely to change to a new permutation sequence.

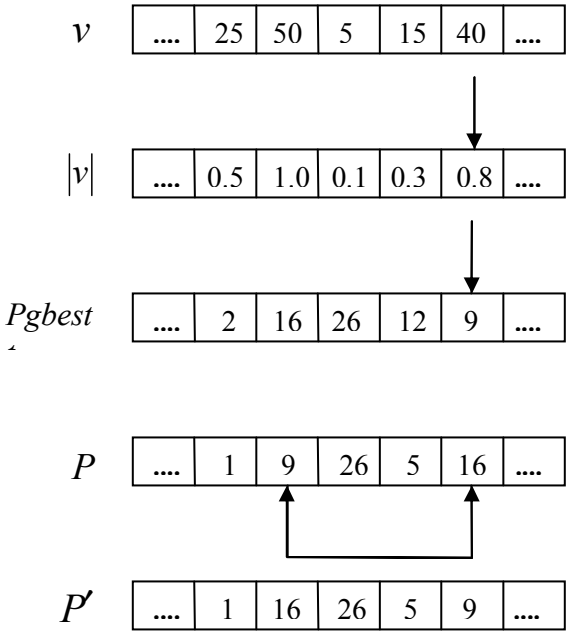

Figure 1: Example for particle position update strategy as described in [17]

Figure 1 shows an example for the particle position update strategy as described in [17]. In Figure 1, *P* and *P'* denote the particle position vector before and after the update respectively. It should be noted that step 4 above is not proposed in [17]. However, our experiments show that this extra step helps in reducing the number of iterations required before a good solution is found.

VI. EXPERIMENTAL RESULTS

Throughout all of our experiments, the number of particles and number of passes were set to 500 and 200 respectively. The rest of the parameters were varied in an ad-hoc way to optimize the results. In order to reduce the number of variables that we need to set by trial and errors, we only investigated the two cases corresponding to $(\lambda_1, \lambda_2) = (1,0)$ and $(\lambda_1, \lambda_2) = (0,1)$.

Figure 2 shows how the average (over 100 randomly selected keys) number of corrected key elements varies with the amount of known ciphertext.

Similarly, Figure 3 shows the percentage of corrected characters versus the amount of known ciphertext. In Figure 4 the error distribution for 100 randomly selected keys are shown when the amount of known ciphertext is 900. The average and variance of the error distribution is 0.04 and 0.0792 respectively.

It is clear that although the bigram statistics provide very good key recovery accuracy, we cannot rely on the unigram for this sort of cryptanalysis.

One should note that, although the value of the cost function improves or at least remains the same, as the number of iterations increases, there are few occasions in which the number of corrected key elements decreases, especially if only a small amount of ciphertext is known. All the results reported in this work are those found after the last iteration.

VII. CONCLUSION

PSO provides a very powerful tool for the cryptanalysis of simple substitution ciphers using a ciphertext only attack.

Given the noticeable accuracy gain of the bigram based attack as compared to the unigram based one; it is interesting to try the trigram in the evaluation function. One may also try to use a cost function that is based on a weight combination of the different n-gram statistics.

One main disadvantage of heuristic optimization techniques (including PSO) is its large sensitivity to parameter variations (e.g., c_1 and c_2 in PSO). 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.

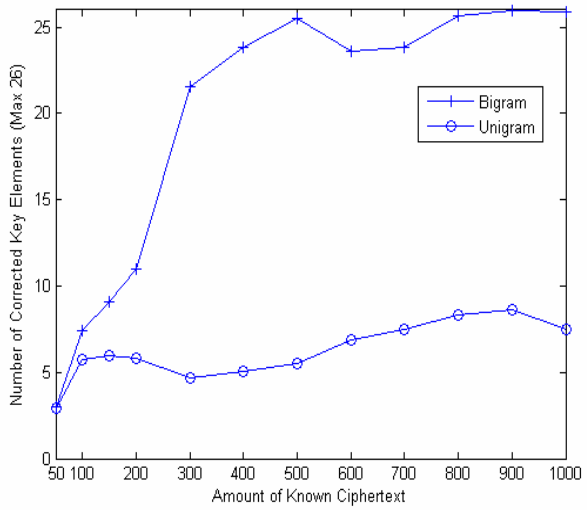

Figure 2: Number of corrected key elements versus the amount of known ciphertext

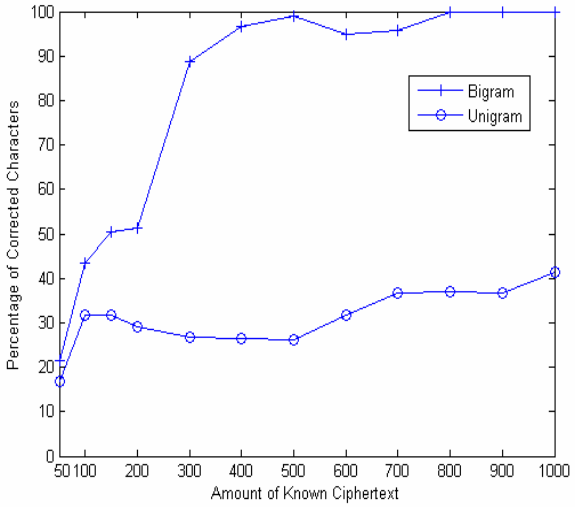

Figure 3: Percentage of corrected characters versus the amount of known ciphertext

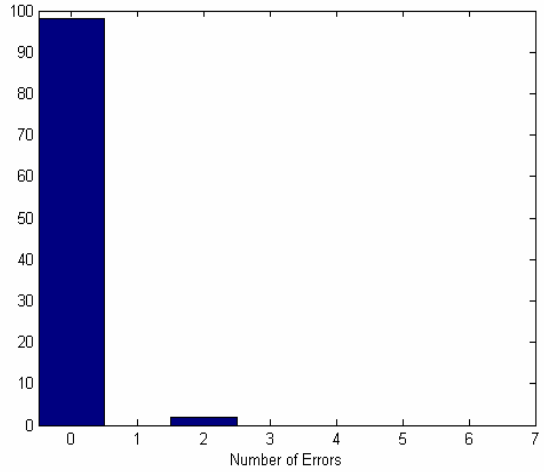

Figure 4: Error distribution (Amount of known ciphertext =900 characters)

REFERENCES

[1] G. D. Kahn, "The Code breakers: the story of secret writing," revised edition, 1996.

[2] Ibraham A. "Al-Kindi: The origins of cryptology: The Arab contributions", Cryptologia, 16(2), pp. 97–126, 1992.

[3] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE int'l conf. on neural networks,* vol. IV, pp. 1942-1948, 1995.

[4] Andrew Clerk, "Optimisation Heuristics for Cryptology*,"* PhD thesis, Queensland University of Technology, 1998.

[5] S. Peleg and A. Rosenfeld, "Breaking substitution ciphers using a relaxation algorithm," *Communications of the ACM*, vol. 22(11), pp.598–605, 1979.

[6] J. Carrol and S. Martin, "The automated cryptanalysis of substitution ciphers," *Cryptologia,* vol.10(4), pp.193–209, 1986.

[7] W. S. Forsyth and R. Safavi-Naini, "Automated cryptanalysis of substitution ciphers," *Cryptologia*, vol.17(4), pp.407–418, 1993.

[8] R. Spillman, M. Janssen, B. Nelson and M. Kepner, "Use of a genetic algorithm in the cryptanalysis of simple substitution ciphers," *Cryptologia,* vol.17(1), pp.31–44, 1993.

[9] D. Bahler and J. King, "An implementation of probabilistic relaxation in the cryptanalysis of simple substitution systems," *Cryptologia,* vol.16(3), pp.219–225, 1992.

[10] M. Lucks, "A constraint satisfaction algorithm for the automated decryption of simple substitution Ciphers," *In Proceedings of CRYPTO'88*, pp. 132–144, 1988.

[11] Thomas Jakobsen, "A fast method for cryptanalysis of substitution ciphers," Cryptologia, pp. 265-274, July 1995.

[12] G. W. Hart, "To decode short cryptograms," *Communications of the ACM,* vol.37(9), pp.102–108, 1994.

[13] 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.

[14] Mohammad Faisal Uddin and Amr M. Youssef, "An Artificial Life Technique for the Cryptanalysis of Simple Substitution Ciphers", to appear in Proc. of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2006), Ottawa, May 2006.

[15] W. Pang, K.Wang, C. Zhou, L. Dong, M. Liu, H. Zhang, and J. Wang, "Modified particle swarm optimization based on space transformation for solving traveling salesman problem," Proc. of 2004 International Conference on Machine Learning and Cybernetics, 2004. vol. 4, 26-29, pp.2342 – 2346, 2004.

[16] L. Cagnina, S. Esquivel, S, and R. Gallard, "Particle swarm optimization for sequencing problems: a case study," Congress on Evolutionary Computation, CEC2004. vol 1, pp. 536 – 541, 2004.

[17] Xiaohui Hu, R.C. Eberhart, and Yuhui Shi, "Swarm intelligence for permutation optimization: a case study of n-queens problem," in *Proc. of the 2003 IEEE on Swarm Intelligence Symposium,* pp.243-246, 2003.

[18] 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,

[19] 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