ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 *2007. No. 7(79)*
ELEKTRONIKA IR ELEKTROTECHNIKA

ELECTRONICS
T170
ELEKTRONIKA

Breaking Transposition Cipher with Genetic Algorithm

R. Toemeh
Department of computer Science and Engineering, Government College of Technology, Coimbatore, India, phone: +919843193540; e-mail: ragtoemeh@yahoo.com
S. Arumugam
Additional Director, Directorate of technical education, Chennai, India

Introduction

In the Brute Force attack the attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained [8]. Many types of classical ciphers exist, although most fall into one of two broad categories: substitution ciphers and transposition ciphers. In the former one, every plaintext character is substituted by a cipher character, using a substitution alphabet, and in the latter one, plaintext characters are permuted using a predetermined permutation.

A simple transposition or permutation cipher works by breaking a message into fixed size blocks, and then permuting the characters within each block according to a fixed permutation, say P. The key to the transposition cipher is simply the permutation P. So, the transposition cipher has the property that the encrypted message contains all the characters that were in the plaintext message. In other words, the unigram statistics for the message are unchanged by the encryption process. The size of the permutation is known as the period. Let's consider an example of a transposition cipher with a period of ten 10, and a key P={7,10,4,2,8,1,5,9,6,3}. In this case, the message is broken into blocks of ten characters, and after encryption the seventh character in the block will be moved to position 1, the tenth moved character in the block will be moved to position 2, the forth is moved to position 3, the second to position 4, the eighth to position 5, the first to position 6, the fifth to the position 7, the ninth to the position 8, the sixth to the position 9 and the third to position 10.

In Table 1 shows the key and the encryption process of the previously described transposition cipher. It can be noticed that the random string "X" was appended to the end of the message to enforce a message length, which is a multiple of the block size. It is also clear that the decryption can be achieved by following the same process as encryption using the "inverse" of the encryption permutation. In this case the decryption key, P^{-1} is equal to {6,4,10,3,7,9,1,5, 8,2}.

Table 1. Example of the transposition cipher key and encryption process

KEY:
Plaintext: 1 2 3 4 5 6 7 8 9 ***10***
Ciphertext:7 ***10*** 4 2 8 1 5 9 6 3
ENCRYPTION:
Position : 123456789***10*** 1234 5678 9*10* 123456789***10***
Plaintext : TRANSPOSITION _ ALGORITHMXXXXXXX
Ciphertext OTNRSTSIPAGI _ OOIARLNXXXHXTXXXM

This paper is organized as follows: A brief description of cryptanalysis, description of attacks on transposition and algorithmic description of the attack on a simple transposition cipher using a genetic algorithm. The results of the genetic algorithm attack are given finally.

Cryptanalysis

A typical cipher takes a clear text message (known as the plaintext) and some secret keying data (known as the key) as its input and produces a scrambled (or encrypted) version of the original message (known as the ciphertext). An attack on a cipher can make use of the ciphertext alone or it can make use of some plaintext and its corresponding ciphertext (referred to as a known plaintext attack) [3].

Cryptanalysis is the process of recovering the plaintext and/or key from a cipher. Many cryptographic systems have a finite key space and, hence, are vulnerable to an exhaustive key search attack. Yet, these systems remain secure from such an attack because the size of the key space is such that the time and resources for a search are not available. A random search through a finite but large key space is not usually an acceptable cryptanalysts tool. In this work, however, we explore the possibility of using a random type search to break a cipher. The focus of this work is on the use of a genetic algorithm to conduct a directed random search of a key space. The ability to add direction to what seems to be a random search is a feature of genetic algorithms which suggests that it may be possible to conduct an efficient search of a large key space [4].

Prior Work

Cryptanalysis of transposition cipher has been carried out by many researchers. Several good papers available in the internet have explained about attack on transposition cipher. Andrew used optimization heuristics in cryptanalysis of transposition cipher [3], using the fitness weight in table 3, as the cost function of genetic algorithm. A.Dimovski and D.Gligoroski use the equation (1) as a cost function [1].

Attacks on the transposition cipher

In this section, we describe Genetic algorithm for attack on the transposition cipher. This section also discusses possible cost functions for the attacks on the transposition cipher.

1) Genetic Algorithms

Brute Force attack has the disadvantage of high computational complexity. In order to overcome this complexity, the Meta heuristic search techniques like Genetic Algorithm are used.

A genetic algorithm is general method of solving problems to which no satisfactory, obvious, solution exists. It is based on the idea of emulating the evolution of a species in nature so the various components of the algorithm are roughly analogous to aspects of natural evolution. Common mathematical tasks amenable to genetic solutions include computing a curve to fit a set of data or approximating NP problems. Often these operators consist of flipping a single random bit of one individual or swapping two randomly selected substrings from a pair of parents to generate a new child. To simulate Darwinian survival of the fittest some representation of the fitness of the individuals must be generated. Genetic algorithm is applied in four steps:

- Initialize algorithm variables: *G* the maximum number of generations to consider, *M* the solution pool size and any other problem dependent variables.
- Generate an initial solution pool containing *M* candidate solutions.
- For *G* iterations, using the current pool:

1. Select a breeding pool from the current solution pool and make pairings of parents.
2. For each parental pairing, generate a pair of children using a suitable mating function.
3. Apply a mutation operation to each of the newly created children
4. Evaluate the fitness function for each of the children
5. Based on the fitness of each of the children and the fitness of each of the solutions in the current pool, decide which solutions will be placed in the new solution pool. Copy the chosen solutions into the new solution pool.
6. Replace the current solution pool with the new one. So, the new solution pool becomes the current one.

- Choose the fittest solution of the final generation as the best solution.

The mutation operation is identical to the solution perturbation technique used in the genetic algorithm attack. That is, randomly select two elements in the child and swap those elements.
The following is flow chart of proposed algorithm; examples for the operations are given in the flow chart:

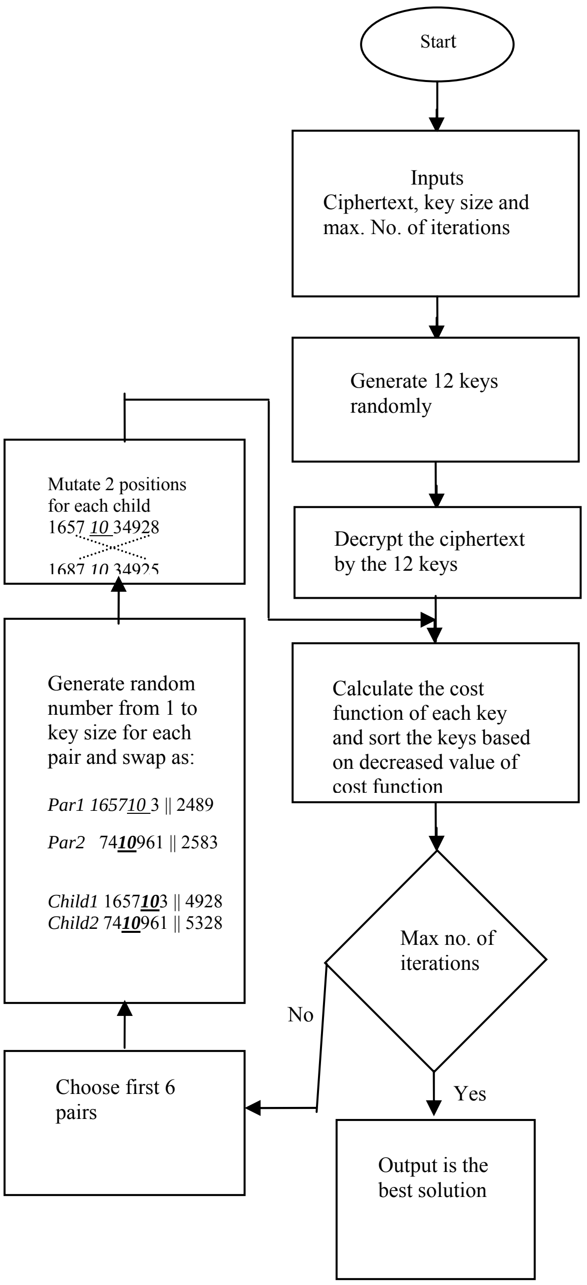
Start
Inputs
Ciphertext, key size and max. No. of iterations
Generate 12 keys randomly
Decrypt the ciphertext by the 12 keys
Calculate the cost function of each key and sort the keys based on decreased value of cost function
Max no. of iterations
No
Yes
Output is the best solution
Choose first 6 pairs
Generate random number from 1 to key size for each pair and swap as:
Par1 165710 3 || 2489
Par2 7410961 || 2583
Child1 1657103 || 4928
Child2 7410961 || 5328
Mutate 2 positions for each child
1657 10 34928
1687 10 34925

Fig. 1. Flow chart of proposed algorithm

Note that for crossover operation the first part of first child is the first part of first parent and second part of first child is the remaining digits as the order of second parent, and the first part of second child is the first part of second parent and second part is the remaining digits as the order of first parent.

2) *Suitability Assessment*

The technique used by A.Dimovski and D.Gligoroski, in [1] to compare candidate keys is to compare n-gram statistics of the decrypted message with those of the language. Equation 1 is a general formula used to determine the suitability of a proposed key (k). Here, A denotes the language alphabet (i.e., for English, [A... Z, _], where _ represents the space symbol), K and D denote known language statistics and decrypted message statistics, respectively, and the indices ***b*** and ***t*** denote the bigram and trigram statistics, respectively. The values of β and γ allow assigning of different weights to each of the two n-gram types.

$$\mathbf{C}^k = \boldsymbol{\beta}\cdot \sum_{i,j\in A}\left|K^b_{(i,j)} - D^b_{(i,j)}\right| + \\ + \boldsymbol{\gamma}\cdot \sum_{i,j,k\in A}\left|K^t_{(i,j,k)} - D^t_{(i,j,k)}\right| \qquad (1)$$

The unigram frequencies for a message are unchanged during the encryption process of a transposition cipher and so, they are ignored when evaluating a key i.e. in equation (1).

The complexity of determining the fitness is O (N^3) (where N is the alphabet size). In the process of determining the cost associated with a transposition cipher key the proposed key is used to decrypt the ciphertext and then the statistics of the decrypted message are then compared with statistics of the language. Matthews proposed an intuitive alternative. Instead of using all possible bigrams and trigrams a subset of the most common ones are chosen [7].

The method of Matthews is to list a number of the most common bigrams and trigrams and to assign a weight (or score) to each of them. Also, the trigram "EEE" was included in the list and assigned a negative score. The idea behind this is interesting. Since E is very common in English, it could be expected that a plaintext message might contain a relatively high number of E's. The same frequency of E's will be present in the ciphertext, but, in this case, it could be expected that, on occasion, three E's might occur simultaneously. Since these never occur normally in the English language, it makes sense to assign such a trigram a negative score. Each weight is applied to the frequency of the corresponding bigram or trigram in the decrypted message.

Table 2 shows the weight table used by Matthews in his paper. The bigrams, trigrams and weights were modified by Andrew John Clark, in [3] to the values shown in Table 3. Notice that the bigram "__" (two consecutive spaces) and the trigram "___" (three consecutive spaces) have been included. Some of the other bigrams and trigrams are slightly different – due to the fact that the space symbol has been included in the encryption in [3] (and was not in the work by Matthews) and also due to different sets of language statistics used.

In this paper we used modified method to evaluate the cost value of the fitness function which is used in [3] and [7], EEE, AND and ING are included to the Andrew table, Table 4 shows the weight table used in this research.

Table 2. The fitness weight table proposed by Matthews.

Bi/trigram	score	Bi/trigram	score
TH	+2	ED	+1
HE	+1	THE	+5
IN	+1	ING	+5
ER	+1	AND	+5
AN	+1	EEE	-5

Table 3. The fitness weight table proposed by Andrew John Clark

Bi/trigram	score	Bi/trigram	score
E_	+2	S_	+1
_T	+1	__	-6
HE	+1	_TH	+5
TH	+1	THE	+5
_A	+1	HE_	+5
___	-10		

Table 4. The fitness weight table proposed in this research

Bi/trigram	score	Bi/trigram	score
EEE	-5	ING	+5
E_	+2	S_	+1
_T	+1	__	-6
HE	+1	_TH	+5
TH	+1	THE	+5
_A	+1	HE_	+5
___	-10	AND	+5

Characteristics of testing environment

The algorithm is implemented using C++ and tested in machine with following configuration: Intel Pentium IV processor with core frequency 3 GHz, RAM 512 MB.

Experimental results

The algorithm has been implemented successfully and the comparison is made upon the amount of ciphertext provided to the attack. These results are presented in Table 5. Here the algorithm was run on different sizes of ciphertext and different number of keys. Here for each size there are some keys have been broken fully. If the ciphertext is having more size the breakable key is more.

Table 5. The amount of key recovered versus available ciphertext for different transposition size

Amount of ciphertext (Letters)	Key recovered							
	Key size(digits)							
	10	**11**	**12**	**13**	**14**	**15**	**16**	**17**
500	10	11	12	10	9	8	7	6
1000	10	11	12	13	14	15	14	14
1500	10	11	12	13	14	15	16	15
2000	10	11	12	13	14	15	16	17

Table 6 shows the required time to break transposition cipher for the cipher text with 1000 letters and different key size.

Table 6. Required time to break the cipher with 1000 letters for keys of different size

Key size	10	11	12	13	14	15
Time (sec)	6.45	22.55	29.31	50	72	100

Conclusion

In the algorithm proposed by A. Dimovski and D. Gligoroski, [1] the recovered key for 1000 letters in ciphertext amount is 13.25 out of 15 key length. In our proposed algorithm the recovered key is 15 for the same amount of ciphertext. So for this size of cipher text and for key size 15 the improvement is 13 percent. The time for breaking the key is less than the time in the Brute Force attack, because the number of possible keys for a transposition cipher with N key size is N! (factorial).

.

References

1. **Dimovski A., Gligoroski D.** Attacks on the Transposition Ciphers Using Optimization Heuristics // International Scientific Conference on Information, Communication & Energy Systems & Technologies ICEST 2003, Sofia, Bulgaria, October 2003.
2. **Andrew Clark, Ed Dawson.** Optimisation Heuristics for the Automated Cryptanalysis of Classical Ciphers // Journal of Combinatorial Mathematics and Combinatorial Computing. – 1998. – Vol. 28. – P. 63–86.
3. **Andrew John Clark.** Optimisation Heuristics for Cryptology / PhD thesis, Information Security Research Centre Faculty of Information Technology Queensland University of Technology. – February 1998.
4. **Ayman M. B. Albassal, Abdel-Moneim A. Wahdan.** Genetic Algorithm Cryptanalysis of a Fiestel Type Block Cipher // International Conference on Electrical, Electronic and Computer Engineering, Egypt, 2004. – P. 217–221.
5. **Goldberg D. E.** Genetic Algorithms in Search, Optimization and Machine Learning. – Delhi, India: Pearson Education, 2004.
6. **Giddy J. P., Safavi-Naini R.** Automated Cryptanalysis of Transposition Ciphers // The Computer Journal. – 1994. – Vol. 37, No. 5. – P. 429–436.
7. **Robert A. J. Matthews.** The use of genetic algorithms in cryptanalysis // Journal of Cryptologia. – April 1993. – Vol 17, No. 2. – P. 187–201.
8. **William Stallings.** Cryptography and Network Security, Principles and Practices, 3rd edition. – Pearson Education, 2004.

Submitted for publication 2007 04 24

R. Toemeh, S. Arumugam. Breaking Transposition Cipher with Genetic Algorithm // Electronics and Electrical Engineering. – Kaunas: Technologija, 2007. – No. 7(79). – P. 75–78.

The aim of the research presented in this paper is to investigate the use of genetic algorithm in the cryptanalysis of transposition cipher. The applicability of genetic algorithms for searching the key space of encryption scheme is studied. The frequency of bigram and trigram is used as an essential factor in objective function. Ill. 1, bibl. 8 (in Lithuanian; summaries in English, Russian and Lithuanian).

Р. Тоемег, С. Арумугам. Дешифрование шифра транспозиции применяя генетический алгоритм // Электроника и электротехника. – Каунас: Технология, 2007. – № 7(79). – С. 75–78.

Цель исследования, представленного в этой статье состоит в том, чтобы исследовать использование генетического алгоритма в криптоанализе шифра транспозиции. Изучена возможность применять генетические алгоритмы для поиска ключевых мест в схемах шифрования. Частота биграмм и триграмм используется как существенный фактор в функции цели. Ил. 1, библ. 8 (на литовском языке; рефераты на английском, русском и литовском яз.).

R. Toemeh, S. Arumugam. Transpozicijos šifro iššifravimas taikant genetinį algoritmą // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2007. – Nr. 7(79). – P. 75–78.

Straipsnyje pateikto tyrimo tikslas – ištirti genetinio algoritmo taikymą analizuojant transpozicijos šifrą. Ištirtos genetinio algoritmo taikymo, ieškant esminių šifravimo schemos taškų, galimybės. Bigramų ir trigramų dažnis panaudotas kaip esminis tikslo funkciją lemiantis veiksnys. Il. 1, bibl. 8 (lietuvių kalba; santraukos anglų, rusų ir lietuvių k.).