Attacks on the Transposition Ciphers Using Optimization Heuristics

A. Dimovski[1], D. Gligoroski[2]

Abstract – **In this paper three optimization heuristics are presented which can be utilized in attacks on the transposition cipher. These heuristics are simulated annealing, genetic algorithm and tabu search. We will show that each of these heuristics provides effective automated techniques for the cryptanalysis of the ciphertext. The property which make this cipher vulnerable, is that it is not sophisticated enough to hide the inherent properties or statistics of the language of the plaintext.**

Keywords – **Transposition substitution cipher, Cryptanalysis, genetic algorithm, simulated annealing, tabu search**

I. Transposition Ciphers

First, we will describe a simple transposition cipher. A 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 i.e. the ciphertext 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 six 6, and a key $P = \{4, 2, 1, 5, 6, 3\}$. In this case, the message is broken into blocks of six characters, and after encryption the fourth character in the block will be moved to position 1, the second remains in position 2, the first is moved to position 3, the fifth to position 4, the sixth to position 5 and the third to position 6.

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

KEY:
Plaintext: 123456
Ciphertext: 421563

ENCRYPTION:	
Position:	123456123456
Plaintext:	HOW_ARE_YOUX
Ciphertext	_OHARWO_RUXY

In Table 1 is shown 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 $\{3, 2, 6, 1, 4, 5\}$.

[1]A. Dimovski, Faculty of Natural Sciences and Mathematics, Ss. Cyril and Methodius University Arhimedova b.b., PO Box 162, 1000 Skopje, Macedonia adimovski@ii.edu.mk 2

[2]D. Gligoroski, Faculty of Natural Sciences and Mathematics, Ss. Cyril and Methodius University Arhimedova b.b., PO Box 162, 1000 Skopje, Macedonia gligoroski@yahoo.com

II. Attacks on the Transposition Cipher

In this section, we will describe three optimization heuristics for attacks on the transposition cipher. Also, a method of assessing intermediate solutions, in the search for the optimum, is discussed.

A. Suitability assessment

The technique used to compare candidate keys is to compare n-gram statistics of the decrypted message with those of the language (which are assumed known). 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.

$$C_k = \beta \sum_{i,j \in A} |K^b_{(i,j)} - D^b_{(i,j)}| + \gamma \sum_{i,j,k \in A} |K^t_{(i,j,k)} - D^t_{(i,j,k)}|. \quad (1)$$

As I said above, 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.

In attacks, which are proposed here, we will use assessment function based on bigram statistics only. The basic reason for this, it is an expensive task to calculate the trigram statistics. The complexity of determining the fitness is $O(N^3)$ (where N is the alphabet size) when trigram statistics are being determined, compared with $O(N^2)$ when bigrams are the largest statistics being used.

B. Simulated annealing attack

In this section an attack on the transposition cipher using simulated annealing is presented.

Simulated annealing is based on the concept of annealing. In physics, the term annealing describes the process of slowly

cooling a heated metal in order to attain a minimum energy state.

The idea of mimicking the annealing process to solve combinatorial optimization problems is attributed to Kirkpatrick et al [4]. The algorithm is (usually) initialized with a random solution to the problem being solved and a starting temperature. The choice of the initial temperature, T_0 is such that $T \gg \Delta E$. At each temperature a number of attempts are made to perturb the current solution. For each proposed perturbation is determined the change in the cost ΔE. And then, if $\Delta E < 0$ then the proposed perturbation is accepted, otherwise it is accepted with the probability indicated by Metropolis Equation 2 which makes a decision based on this cost difference and the current temperature.

$$\mathrm{Pr}obability(E_1 \Rightarrow E_2) = \exp\Big(-\frac{\Delta E}{T}\Big). \qquad (2)$$

If the proposed change is accepted ($\Delta E < 0$ or Probability $(E_1 \Rightarrow E_2) > 0.5$) then the current solution is updated. Generally, the temperature is reduced when either there a predefined limit in the number of updates to the current solution has been reached or after a fixed number of attempts have been made to update the current solution. The algorithm finishes either when no new solutions were accepted for a given temperature, or when the temperature has dropped below some predefined limit.

Here, in this attack, the Equation 1 is utilized when determining the cost of the solutions, and candidate solutions are generated from the current solution by swapping two randomly chosen positions. Description of this algorithm is given in Fig. 1.

This simulated annealing was implemented and the experimental results are given below in Section 3, which compare this technique with the genetic algorithm attack described in Section 2.3 and the tabu search attack, which is described in Section 2.4.

1. Inputs to the algorithm are the intercepted ciphertext, the key size (permutation size or period) P, and the bigram statistics of the plaintext language.
2. Initialize the algorithm parameters: the maximum number of iterations MAX, the initial temperature T_0, and the temperature reduction factor T_{FACT} .
3. Set T = T_0 and generate a random initial solution K_{CURR} and calculate the associated cost C_{CURR}.
4. For i = 1, . . . MAX, do:
   (a) Set N_{SUCC} = 0.
   (b) Repeat 100 · P times:
      i. Generate a new candidate key K_{NEW}:
         A. Choose n_1,$n_2 \in$ [1, P], $n_1 \neq$ n2.
         B. Swap n_1 and n_2 in K_{CURR} to create K_{NEW}.
      ii. Calculate the cost C_{NEW} of K_{NEW}. Find the cost difference $\Delta E = C_{NEW} - C_{CURR}$ and consult the Metropolis criterion, Equation 2 to determine whether the proposed transition should be accepted.
      iii. If the transition is accepted set $K_{CURR} = K_{NEW}$ and $C_{CURR} = C_{NEW}$ and increment N_{SUCC}.
   (c) If $N_{SUCC} > 10 \cdot P$ go to step 4d.
   (d) If $N_{SUCC} = 0$ go to Step 5.
   (e) Reduce T (T = T · T_{FACT}).
5. Output the current solution K_{CURR}.

Fig. 1. Simulated annealing attack on the transposition cipher

C. Genetic algorithm attack

The genetic algorithm is more complicated than the simulated annealing attack. This is because a pool of solutions is being maintained, rather than a single solution. An extra level of complexity is also present because of the need for a mating function and for a mutation function.

In this attack also, the Equation 1 is utilized when determining the cost of the solutions. The mating i.e. reproduction technique used here for creating the two children is now given:

1. Notation: p_1 and p_2 are the parents, c_1 and c_2 are the children, $p_i(j)$ denotes the element j in parent i, $c_i(j)$ denotes element j in child i, $\{C_i^{j,k}\}$ denotes the set of elements in child i from positions j to k with the limitation that if $k = 0$ or $j = P + 1$ then $\{C_i^{j,k}\} = \{\emptyset\}$,

2. Child 1:

(a) Choose a random number $r \in [1, P]$
(b) $c_1(j) = p_1(j)$ for $j = 1, ..., r$
(c) For $i = 1, ..., P - r$ and $k = 1, ..., P$
If $p_2(k) \notin \{C^{1,i+r-1}\}$
then
$- c_1(i + r) = p_2(k)$
else $k = k + 1$

3. Child 2:

(a) Choose a random number $r \in [1, P]$
(b) $c_2(j) = p_1(j)$ for $j = P, ..., r$
(c) For $i = 1, ..., r$ and $k = P, ..., 1$
If $p_2(k) \notin \{C^{r-i+1,P}\}$
then
$- c_2(r - i) = p_2(k)$
else $k = k - 1$

The mutation operation is identical to the solution perturbation technique used in the simulated annealing attack. That is, randomly select two elements in the child and swap those elements.

In Fig. 2 is an algorithmic description of the attack on a simple substitution cipher using a genetic algorithm.

The results of the genetic algorithm attack are given in Section 3.

D. Tabu search attack

The transposition cipher can also be attacked using a tabu search. This attack is similar to the simulated annealing one with the added constraints of the tabu list. The same perturbation mechanism i.e. swapping two randomly chosen key elements is used to generate candidate solutions. In each iteration the best new key found replaces the worst existing one in the tabu list. The overall algorithm is described in Fig. 3.

The experimental results obtained using all three attacks are presented in the next Section.

1. Inputs to the algorithm are the intercepted ciphertext, the key size (permutation size or period) P, and the bigram statistics of the plaintext language.
2. Initialize the algorithm parameters: the solution pool size M, and the maximum number of iterations MAX.
3. Generate an initial pool of solutions (randomly) P_{CURR}, and calculate the cost of each of the solutions in the pool using Equation 1.
4. For i = 1, . . . ,MAX, do:
   (a) Select M/2 pairs of keys from P_{CURR} to be the parents of the new generation.
   (b) Perform the mating operation described above on each of the pairs of parents to produce a new pool of solutions P_{NEW}.
   (c) For each of the M children perform a mutation operation described above.
   (d) Calculate the cost associated with each of the keys in the new solution pool P_{NEW}.
   (e) Merge the new pool P_{NEW} with the current pool P_{CURR}, and choose the best M keys to become the new current pool P_{CURR}.
5. Output the best solution from the current key pool P_{CURR}.

Fig. 2. Genetic algorithm attack on the transposition cipher

1. Inputs to the algorithm are the intercepted ciphertext, the key size (permutation size or period) P, and the bigram statistics of the plaintext language.
2. Initialize the algorithm parameters: the size of the tabu list S_TABU, the size of the list of possibilities considered in each iteration S_POSS, and the maximum number of iterations to perform MAX.
3. Initialize the tabu list with random and distinct keys and calculate the cost associated with each of the keys in the tabu list.
4. For i = 1, . . . ,MAX, do:
   (a) Find the best key i.e. the one with the lowest cost in the current tabu list, K_{BEST}.
   (b) For j = 1, . . . S_POSS do:
      i. Apply the perturbation mechanism described in the simulated annealing attack to produce a new key K_{NEW}.
      ii. Check if K_{NEW} is already in the list of possibilities generated for this iteration or the tabu list. If so, return to Step 3(b)i.
      iii. Add K_{NEW} to the list of possibilities for this iteration.
   (c) From the list of possibilities for this iteration find the key with the lowest cost, P_{BEST}.
   (d) From the tabu list find the key with the highest cost, T_{WORST}.
   (e) While the cost of P_{BEST} is less than the cost of T_{WORST}:
      i. Replace T_{WORST} with P_{BEST}.
      ii. Find the new P_{BEST}.
      iii. Find the new T_{WORST}.
5. Output the best solution from the tabu list K_{BEST}.

Fig. 3. Tabu search attack on the transposition cipher

III. Experimental Results

The three techniques were implemented in Java as described above and a number of results were obtained.

The first comparison is made upon the amount of ciphertext provided to the attack. These results are presented in Table 2. Here each algorithm was run on differing amounts of ciphertext – 50 times for each amount. The results in Table 2 represent the average number of key elements correctly placed for a key size of 15 in this case. Note that because a transposition cipher key, which is rotated by one place, will still properly decrypt a large amount of the message, a key element is said to be correctly placed if its neighbors are the same as the neighbors for the correct key (except for end positions). In that case, the message will almost certainly still be readable, especially if the period of the transposition cipher is large. It can be seen from the results that each of the three algorithms performed roughly equally when the comparison is made based upon the amount of known ciphertext available to the attack.

Table 2. The amount of key recovered versus available ciphertext, transposition size 15

Amount of ciphertext	SA	GA	TS
200	5	7	4.75
400	10.75	11.5	9.25
600	11.25	12.5	11.5
800	12.5	13	12.25
1000	12.75	13.25	13

Table 3 shows results for the transposition cipher based on the period. It should be noted that for period less than fifteen, with one thousand available ciphertext characters, each of the algorithms could successfully recover the key all the time. The table shows that the simulated annealing attack was the most powerful. For a transposition cipher of period 30 the simulated annealing attack was able to correctly place 25 of the key elements, on the average.

Table 3. The amount of key recovered versus transposition size, 1000 known ciphertext characters

Transposition size	SA	GA	TS
15	12.75	13.25	13
20	16.5	17	16.75
25	20.15	21.5	21
30	25	25.25	25.5

References

[1] Fred Glover, Eric Taillard, and Dominique de Werra. A users guide to tabu search. Annals of Operations Research, 41:328, 1993.

[2] D.E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, Massechusetts, 1989.

[3] Robert A. J. Matthews. The use of genetic algorithms in cryptanalysis. Cryptologia, 17(2):187201, April 1993.

[4] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. Science, 220(4598):671680, 1983.