A Powerful Genetic Algorithm to Crack a Transposition Cipher

Omar Alkathiry and Ahmad Al-Mogren

***Abstract*—Security of information systems depends heavily on the strength of the cryptosystem. Throughout the years, several cryptosystem algorithms have been developed. These algorithms may inherit some weakness that can jeopardize the integrity of the data. In this paper, we present a genetic algorithm to crack a transposition cipher that extends on the notable research in the field and introduces new ideas including a novel crossover function, a dictionary of the most used words in the English language to evaluate the fitness of the keys in any generation, a dynamic selection method, and a variable generating number. This algorithm starts with a very small randomly selected set of keys, and proceeds the crossover operation on the highly fit keys to produce next generations until a specific number of generations, the final result produces a key that is either a perfect match to the original encryption key or one that is very close. Our experiments and results show mostly optimal solutions for the keys in linear time performance which is a dramatic improvement to the brute force algorithm that takes a factorial time to crack the key.**

***Index Terms*—Cipher key, cryptanalysis, dictionary matching, genetic algorithm, transposition cipher.**

I. Introduction

The world of computer security has had a great deal of development in creating method to protect the computer systems from threats, these threats come from several areas concerning authentication, authorization, confidentiality, and non-repudiation.

One of the pillars of computer security is the encryption of data, since networks are shared between millions of people; and thus transmitted data can be viewed and intercepted, the network infrastructure could not be trusted to send plain data, transferred data has to be interpreted by only the ones who are authorized to; others should not be able to under-stand the content, this is usually done by encryption, the process of changing the content of the text according to an algorithm by using keys that are normally shared by the interested parties.

Cryptanalysis is the science concerned with analyzing the cipher text trying to identify patterns to reveal the encryption algorithm used, extract the key, and extract the plain text; thus expose the encryption. There has been sever-al techniques to achieve this goal for different encryption algorithms, manual cryptanalysis is a very time consuming process and is not a desired solution in cryptanalysis, hence; several automated techniques were developed in this area.

The key size used in encryption is always recommended to be long to make exhaustive search very difficult, this alongside with advances in encryption algorithms makes cryptanalysis and breaking encryption algorithm a challenging task.

A classical symmetric encryption method is the transposition encryption where the plain text is organized into columns and rows, and shuffled based on a key, the order of the columns is based on the character code of the letters, and the recipient uses this key to decrypt the content. Fig. 1 describes this encryption.

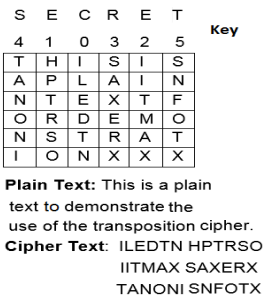

Fig. 1. A Demonstration of a simple transposition cipher.

This paper introduces a new algorithm to crack a columnar transposition cipher in a linear time, several methods were suggested in the literature that provide algorithms to crack the transposition cipher encryption, including the Ants Colony algorithm, the Simulated Annealing algorithm, the Tabu Search, and other genetic algorithms, we show that our method provides better results in terms in performance and accuracy.

The paper is organized as follows: first, we talk about the previous work in Section II. In Section III, we provide detailed description of the proposed algorithm. In Section IV, we show the experiments and results. Finally, we talk about the conclusions found, and the future work needed to enhance this algorithm in Section V.

II. Related Work

There has been several research to develop methods to crack the transposition cipher, some of them used simulated annealing [1], that is a local search where an evaluating function is used to assess the current state and tries to move to better states in the search space, allowing some bad moves in the process to insure that the algorithm does not get stuck in a local optimum option, the simulated annealing method is

Manuscript received January 21, 2014; revised May 16, 2014. This work was supported by the Research Center of College of Computer and Information Sciences, King Saud University RC130311. The authors are grateful for this support.

Omar Alkathiry is with King Saud University, Riyadh, Saudi Arabia (e-mail: omar.alkathiry@gmail.com).

Ahmad Al-Mogren is with College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia (e-mail: ahalmogren@ksu.edu.sa).

inspired by the process of annealing metals, where the heated metal is slowly heated to achieve a better consistency "Energy", the experiments made using this method to crack the transposition cipher as described by the authors proves it to be an effective algorithm; however does not match the performance of using a genetic algorithm, the clear advantage that simulated annealing has over other methods is the simplicity of the algorithm.

In an interesting research using particle swarm optimization (PSO)[2] that was used to crack a transposition cipher, this method is an optimization method based on the swarm intelligence, inspired by the flocks of flying birds and other swarms like salmon fish, PSO is used to solve this problem by first choosing a random set of keys, each called a particle, all of those keys store the position of the best solution found so far, the particles move and if a better solution is found the particle updates its best solution until a better solution is found, the experiments using this method as described by the authors produced very good results; the time taken to crack a ten digit key of a 250 characters cipher was only 1.3 seconds.

Other researchers used Tabu search [3], In this algorithm the best solution is initialized with a random key, this key is going to be scored, then a list of keys created by swap-ping number in this keys is created and the new set is scored, the best of these keys will replace the previous one and the process will be repeated for a specific number of times, the Tabu algorithm also has very good results but are not as good an the genetic algorithm, the advantage of this method is finding the best key in a faster time, but the final result will take longer time because other keys will be tested.

Other research used the Ant Colony algorithm [4], this method is inspired by the ant behavior and the pheromones they produce in order to for the colony to take the best path to reach a specific goal; however this method performs poorly if compared to the GA, SA, and Tabu search.

Finally we will consider other methods that use genetic algorithm to crack the transposition cipher, a good research by Toemeh R. *et al.* [5] uses a genetic algorithm, which our work extends on, produces very good result, however our view is that there are several issues and weaknesses in this research, where the authors did not explain nor give a good reason for the crossover operation and justify why would it improve the next breed, in addition this method fixed the number of generations to test, and mutated all the options, another weakness we observed in this algorithm is that the selection of the initial population was not explained thoroughly when it ought to; since the search space is huge, (Factorial of the key length).

Another research using a genetic algorithm approach by M Heydari *et al*., [6], approached the problem in similar manner as the previous research in addition to making more explanations on the mutation and the crossover operation, the initial population is dynamic in this research; however the number of generations was not discussed where we think it is an essential factor in any genetic algorithm.

There are other genetic algorithms used in this field and most of them have an unrealistic assumption that the cipher text will include spaces [5], [6], this assumption produces unreliable results because the fitness of the key is based on the how many matches found after trying decrypting using any given key; the score increases when there are specific bi-grams or tri-grams, and most of them contain space, as Fig. 2.

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

Fig. 2. Scoring table used by R. Tomeh.

We view that all these factors as weaknesses that we will over-come in our research.

III. Proposed Method Using a Genetic Algorithm and Dictionary Matching

At this section we will provide a detailed explanation of the method we are proposing, Fig. 3 shows the outline of the steps in the algorithm.

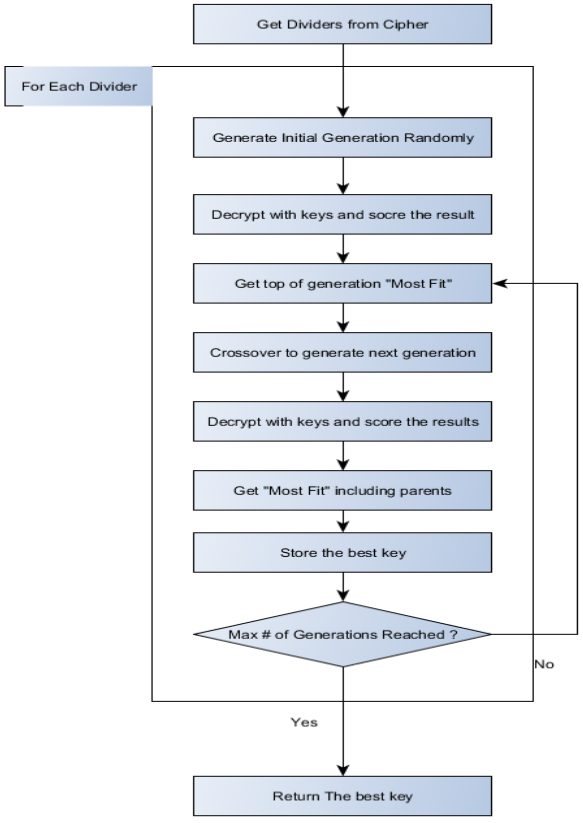

Fig. 3. The steps of the algorithm.

This algorithm consists of the following steps: first, from the cipher text find out all the possible numbers that divide the length of the text evenly, this is done to get all the possible lengths of the key used in encryption, then for each divider generate a random set of keys that will be the starting generation, the number of possibilities for the key in this length is the factorial of length of the key; thus a very small number of keys has to be considered in a fare and meaningful way, after that we use these keys to decrypt the cipher and score the results against a dictionary of frequently used words, the scoring operation finds the number of matches found in

the decryption results from the dictionary list, after that we get the top scorers "Highly fit options" and use them to generate the next generation of nodes by using the crossover operation, the crossover operation also has to be meaningful with the aim to produce a more fit generation, the number of generations depends on the key length; for example a key of four digits will have less generations than a key with ten digits and so on, the formula used to generate the number of generations to go through is the following:

$$G = G\,(K\text{-}1) \times 1.5$$

where G is the number of generations, and K is the length of the key considered, starting with $G = 4$ for the key length 5, this formula is the result of experimentation, increasing the number of generations increases the accuracy; however also increases the time, we have found that this formula give us constant accuracy for 10 attempts to be from 65% - 100% so at least one of the results will be very easily readable if not perfect., at the end of testing all the keys from all the dividers in each generation, the algorithm returns the best scoring key, as the most probable key used in the encryption, the results are mostly a perfect match or a very close one.

This was a brief of the steps and the following details these steps.

Step 1: Get Cipher Text Dividers

After acquiring the cipher text, in this step the length of the cipher text is divided by all the numbers from 2 until the parameterized key length, since the transposition cipher is a block cipher; the cipher text will always construct a rectangle, that is rows and columns, the length of the row is the length of the key and the length of columns is the cipher length divided by the key length, and the extra slots will be filled 'padded' with 'X's or any other character to make the rows with the same length as we saw in Fig. 1, for example if the length of the cipher text is: 224 then these are possible key length in the encryption process given that the maximum specified is 10: 2,4,7,8. These numbers divided 224 evenly.

Step 2: Generate the Start Generation

Randomly generate N keys to act as the first generation, we use the random generation as a quick way to create valid keys, the number of these keys is going to be relative to the key length, according to the following formula:

$$N = K \times 50$$

where N is the number of keys and K is the key length.

After we get the number of starting keys; randomly generate N keys with length K that will be the starting generation for our algorithm, for each index in the key a random number will be assigned that is within the key length. For example if the key length is 6 we could start with a random option for index 0 from (0-5) for example, if the random option was 4, then index 1 will have the options (0, 1, 2, 3, 5,) and so on until we generate a valid key, this process will be repeated for all the keys, the key will be neglected if in the rare case the key is repeated.

Step 3: Execute the Genetic Algorithm

In this step, for each of the dividers perform the following:

- Decrypt the cipher text using keys in the starting generation, and score the results based on the number of matches found in the plain text against a dictionary of most frequently used English words [7] scoring each higher according to their frequency, so for example the word "the" will have a higher score than the word "most", an addition of giving minus scores if certain patterns were encountered such as three same consecutive letters.
- Select a small percentage of this generation with the highest scores "Most Fit" depending on the key size and use them in the crossover operation to participate in the next generation, note that these keys could survive the next generation and be used in another crossover loop to create more keys if their score is high enough, this feature corresponds more naturally to the genetic operation where ancestors could live longer than descendants if they are more fit or if the new child is weak.
- Perform the crossover operation which described by Fig. 4.

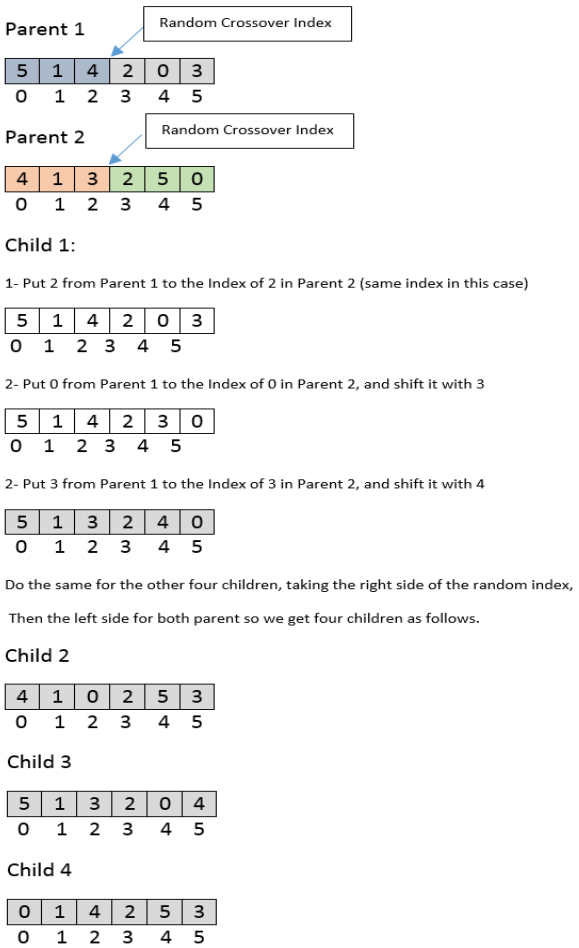

Fig. 4. The crossover operation.

- In this operation, for each pair of keys, called parents here, and ordered by their fitness, so the most fit is going to "marry" the second most fit and so on, first we randomly select an index in the first parent $P1$, we call this the "crossover index", then the numbers in the cells with index larger than crossover index are switched with the numbers in indices of the same value in the other parent, then the same process will be repeated for the number in indices lower that the crossover index to

produce the second child, this process is repeated on each parent and for both sides of the selected index to generate four children. The novelty here is that this crossover operation tries to get better children by merging two features of the parents, so the index of numbers at the fit parent P1, could be better if replaced with the index of the same number in another fit parent P2 and so on this way we are taking two fit parents and merging their properties to generate possibly better offspring.

- Perform the mutation operation which is defined by randomly selecting two indices in the key and swapping their values.
- From the new generation that includes the parents and their children perform the fitness operation by scoring this population against the dictionary, and extract the top percentage of the new population then perform the crossover operation, to produce next and probably better generation of keys, note that parents can survive more the one generation if they always score high.
- Repeat steps from 2 to 4 until we go through all the keys in all the generations, and always test to see if a better key was tested, otherwise keep the highest one so far, an addition if a key survives a generation, we use the same score found before and thus saving processing time.
- At the end a key with the highest score amongst all the dividers and all the generations is selected as the best possible key.

IV. Experiments and Results

A Java application was developed to test this algorithm, several experiments were performed on different texts with lengths (200 – 500 – 1000) characters, and the results of cracking the key were recorder, the keys used to encrypt the plain text were random keys of length from 5 to 20, the accuracy as described earlier is very high and mostly optimal, otherwise; the key would be very close the original encryption key and the plain text would be understandable.

One of the experiments we made using our algorithm was aimed to find the average time to crack the encryption key using different cipher text lengths, we made 10 experiments on each key length and Table I shows the average time needed to crack the encryption key with different text sizes, the average accuracy of the results if from 65% - 100%, which means at least of the 10 results will be acceptably readable.

TABLE I: Time Taken to Crack the Cipher

Time Taken (Seconds)		Key Size										
		10	11	12	13	14	15	16	17	18	19	20
Text Size	200	0.284	0.239	0.280	0.279	0.315	0.703	0.804	0.767	0.917	0.917	1.687
	500	0.612	0.636	0.741	0.825	0.855	1.806	1.975	2.080	2.233	2.200	3.724
	1000	2.588	2.857	2.938	3.456	3.482	7.326	7.881	8.872	9.034	9.604	12.798

The performance of our algorithm can be demonstrated in the Fig. 5.

Another experiment we made was aimed to find the amount of key successfully recovered, The criteria we used is the amount of key recovered for the given cipher text size, given that an index of a number in the key is considered correct if the neighbors of this number are the same except the first and last index, in addition of course to being at the same index, the results of this experiments are shown in Table II.

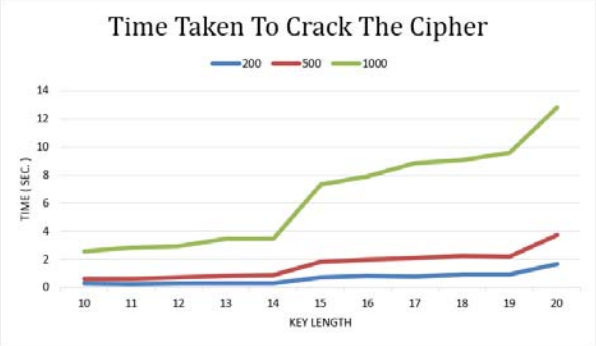

Fig. 5. Algorithm performance.

TABLE II: Amount of Key Recovered

Cipher Text Length	Amount of Recovered Key Size										
	10	11	12	13	14	15	16	17	18	19	20
200	10	11	12	13	14	15	16	17	17	19	18
500	10	11	12	13	14	15	16	15	15	13	12
1000	10	11	12	13	13	13	12	11	10	10	11

We made a comparison between our results with the results of R.Tomeh [5], Fig. 6 demonstrated this comparison.

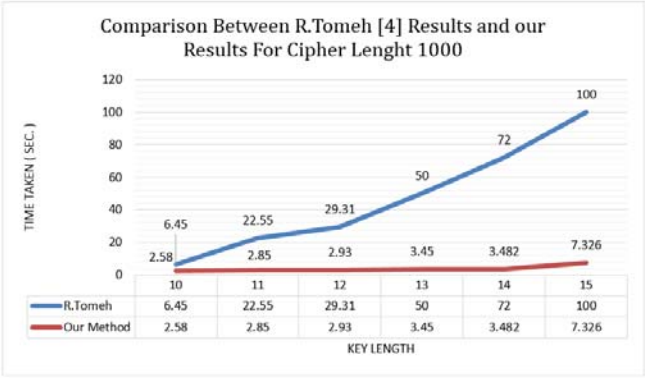

Fig. 6. Comparison with R.Thomeh [5] results.

These results show that this algorithm performs faster and has much smaller growth rate function than the fastest algorithm so far [5] moreover the genetic algorithm approach have been proved to be the best and faster performing method to crack a transposition cipher [8].

The application was developed so that the results could be tweaked, and the user could change some of the parameters to achieve better results, these parameters are: the number of generations, the mutation frequency, and the initial pool percentage.

V. Conclusions and Future Work

In this paper, we introduced a new algorithm to crack the transposition cipher in linear time, our algorithm was tested against several other algorithms in the same field and proved to be better in performance in at least 50%, the main key feature in our algorithm is the novel crossover operation that helped in producing better keys and reaching the optimal key

in a very short time and skim through the huge search space in a very fast and efficient manner, another advantage is the use of a dictionary of most common words in the English language in contrast to the bi-gram and tri-grams method used heavily in other research, this algorithm has one of the features that was exclusive in the Tabu search which is that the keys stay in the pool of keys and survive generations until better keys are introduced in the pool of highly fit keys[3]. The last distinct feature of this algorithm is that several factors that affect the genetic algorithm are de-pendent on the key size which makes it produce the same quality of result for all key lengths.

For future work we are going to test our algorithm on longer key lengths, and we also are going to try improve the performance of some parts of the algorithm through experimentation, another area that could be explored is testing the same algorithm on the bi and tri-gram matching method used in similar papers, a possible improvement at the performance of the algorithm is the usage of parallelization, since the key testing do not require to be sequential and there is no dependency between several parts of the algorithm, parallelizing them will certainly improve the performance..

REFERENCES

[1] A. Dimovski and D. Gligoroski, "Attacks on the transposition ciphers using optimization heuristics," in *Proc. the International Conference on Environmental Science and Technology*, 2003, pp. 1-4.

[2] S. M. Hameed and D. N. Hmood, "Particles swarm optimization for the cryptanalysis of transposition cipher," *Journal of Al-Nahrain University*, vol. 13, no. 4, December, 2010, pp. 211-215

[3] A. K. Verma, M. Dave, and R. C. Joshi, "Genetic algorithm and tabu search attack on the mono-alphabetic subsitution cipher in adhoc networks," *Journal of Computer Science*, vol. 3, no. 3, 2007.

[4] M. D. Russell, J. A. Clark, and S. Stepney, "Making the most of two heuristics: Breaking transposition ciphers with ants," in *Proc. 2003 Congress on the Evolutionary Computation,* vol. 4, 2003.

[5] R. Toemeh and S. Arumugam, "Breaking transposition cipher with genetic algorithm," *Electronics and Electrical Engineering*, vol. 79 2007.

[6] M. Heydari, L. S. Gholamreza, and M. H. Mohammad, "Cryptanalysis of transposition ciphers with long key lengths using an improved genetic algorithm," *World Applied Sciences Journal*, vol. 21, no. 8, 2013.

[7] Oxford English Dictionary. (2013). [Online]. Available: http://www.oxforddictionaries.com/words/the-oec-facts-about-the-language.

[8] P. Garg, "Genetic algorithms, tabu search and simulated annealing: a comparison between three approaches for the cryptanalysis of transposition cipher," *Journal of Theoretical & Applied Information Technology*, vol. 5, no. 4, 2009.

Omar Alkathiry was born in Saudi Arabia, 1986. He received a bachelor's degree in information systems in 2008, and a master's degree in computer science in 2014, both from the College of Computer and Information Sciences of King Saud University in Riyadh, Saudi Arabia.

He has worked as a programmer analyst, then as a java developer, followed by a software engineer, and now he works as a senior software engineer at Technology Control Company, Riyadh, Saudi Arabia.

Ahmad Al-Mogren obtained his PhD in computer sciences from Southern Methodist University, Dallas, Texas, USA in 2002. Previously, he worked as an assistant professor of Computer Science and the head of the Scientific Council at Riyadh College of Technology. He also served as the dean of Computer College and the head of the Council of Academic Accreditation at Al Yamamah University.

Presently, he works as the vice dean for quality and development at King Saud University in Saudi Arabia. He has taught several courses such as computer networks, mobile computing, databases, digital communication, clustering and fault tolerance. His research areas of interest include networking, mobile computing, computer security and data consistency.