ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20190314095054
Original scientific papers

Cryptanalysis of Polyalphabetic Cipher Using Differential Evolution Algorithm

Arkan Kh Shakr SABONCHI, Bahriye AKAY

Abstract: Today it is necessary to keep information secure and cryptography is the most common technique for data security. The Vigenere cipher, one of the polyalphabetic encryption algorithms, has been used in the history by substitution of the plaintext letters with other alphabet letters using a secret keyword and a systematic table. In order to make the ciphertext readable with a keyless procedure, the cryptanalysis technique is used. However, extracting all possible permutations of the letters is exhaustive or frequency analysis is ineffective to extract the letters from the cipher. Therefore, this study aims to propose an efficient polyalphabetic Vigenere cipher cryptanalysis using Differential Evolution algorithm on English and Turkish texts at different lengths. The efficiency of the Differential Evolution algorithm is compared to those of Genetic Algorithm and Particle Swarm Optimization algorithms in terms of the number of key letters recovered correctly. The results show that Vigenere cipher analysis using Differential Evolution algorithm is more effective in polyalphabetic cryptanalysis.

Keywords: cryptanalysis; differential evolution; genetic algorithm; particle swarm optimization; vigenere cipher

1 INTRODUCTION

Cryptography and cryptanalysis are two highly common methods used in the cryptology. The first is related with generating a variety of algorithms for encoding and decoding messages to keep them confidentiality secure, whereas in the latter, the goal is to work on the ciphertext to extract the plaintext even though there is no prior information and authorization between the sender and the receiver of the message to retrieve the keyword [1]. Typically, the classical ciphers are classified into two sub-groups: transposition (or permutation) and substitution ciphers. An encoder uses a particular permutation to divide the plaintext into several blocks with a certain size or interchanges the letters in a systematic way [2-5].

The cryptosystems for substitution cipher may also be sub-categorized as monoalphabetic and polyalphabetic. The most common polyalphabetic algorithm is Vigenere cipher [6, 7]; it works through replacement of each plaintext letter with another letter, which is found through addition of the index numbers of the plaintext character and an arbitrarily chosen code word. The original message is encoded using a table of rows and columns formed by alphabet letters in English or Turkish or any other language, through the replacement of the letters in the plaintext with the letters in the table based on the indices [8-10]. For instance, the number of possible keywords is 26^m in English or 29^m in Turkish, where m is the key length. The plaintext letters are re-written as a sequence of integers as well as the key letters. The integer string of the message is split into reasonable blocks, depending on the key size. Eq. (1) and Eq. (2) are used for encryption and decryption purposes, respectively:

$$C_i = \left(P_i + K_i\right) mod\ N \tag{1}$$

$$P_i = \left(C_i - K_i\right) mod\ N \tag{2}$$

where $P = (P_1, P_2, P_3, ..., P_n)$ is a plain text block, $K = (K_1, K_2, K_3, ..., K_n)$ is key, $C = (C_1, C_2, C_3, ..., C_n)$ is ciphertext block and N is the number of alphabet letters in the target language.

Since a potential key is a permutation for each alphabet letter, a wide range of permutations are available for the key. Therefore, manual cryptanalysis or cryptanalysis using Brute force are ineffective due to their computational cost and work. Accordingly, metaheuristic algorithms are useful to make a systematical search and to find the optimal key.

Nature-inspired algorithms have been utilised by the researchers in the cryptanalysis of classical cryptosystems and positive outcomes have been claimed by many researches. Spillman et al. [11] implemented a Genetic Algorithm (GA) to break a Monoalphabetic Substitution Cipher. Furthermore, Genalyst was proposed by Matthews [12] to break the transposition cipher. Clark [13] presented GA, Tabu Search (TS) and Simulated Annealing (SA) to cryptanalyze the substitution cipher. Moreover, Clark et al. [14] were the first to recommend the adoption of GA in order to complete an attack on a polyalphabetic substitution cipher. Clark and Dawson [15] improved the work by a parallel GA to attack the Vigenere cipher. Moreover, Clark and Dawson [16] performed a comparison among SA, GA and TS on simple substitution ciphers. Dimovski and Gligoroski [17] applied SA, GA and TS in order to achieve transposition cipher cryptanalysis. Verma et al. [18] presented a monoalphabetic substitution cipher based on GA and TS and compared the overall efficiency of these algorithms. An automated approach to the cryptanalysis of transposition cipher was developed in the works of Song et al. [19] and Garg [20] based on GA, TS and SA algorithms. In addition, Omran et al. [21] developed a GA to attack the Vigenere Cipher. Bhateja and Kumar [22] adopted elitism in GA with a novel fitness function and applied it to cryptanalyze a Vigenere cipher. In this regard, Boryczka and Dworak [23] considered the evolutionary algorithms to increase the speed of cryptanalysis of the transposition cipher. Uddin and Youssef [24] applied Ant Colony Optimization (ACO) in order to attack simple substitution ciphers. Bhateja et al. [25] investigated the performance of Cuckoo Search (CS) algorithm in the cryptanalysis of the Vigenere cipher, whilst Luthra and Pal [26] directed their efforts towards examining the integration of mutation and crossover with the Firefly Algorithm (FA) for cryptanalysis of the monoalphabetic cipher. Sabonchi and

Akay [27, 28] presented Artificial Bee Colony algorithm (ABC) in cryptanalysis of the substitution ciphers.

Nonetheless, such techniques are inefficient in analysis of the cipher if the key size exceeds 15 characters. One of the successful evolutionary algorithms in problem-solving, Differential Evolution (DE) [29] gained a success on many problems in various research fields [30]. This encourages further work on DE algorithm in the cryptanalysis of Vigenere cipher, which is the aim of this study. Several encrypted English and Turkish texts at different lengths, and keyword sizes are used to evaluate the efficiency of DE, GA and Particle Swarm Optimization (PSO) on Vigenere cipher analysis.

The rest of the paper is organized as follows: in Section 2, some brief descriptions of the algorithms used in this study are presented. In Section 3, the proposed cryptanalysis approach based on DE algorithm is provided, In Section 4, experimental study is explained, and the results are discussed. Finally, Section 5 is dedicated to the conclusion and future work.

2 BRIEF DESCRIPTION OF THE ALGORITHMS USED IN THE STUDY

Metaheuristic algorithms find solutions systematically by directed and randomized searches for the problems especially computationally unmanageable. We used some of these algorithms in our study including Genetic Algorithm, Particle Swarm Optimization and Differential Evolution. Brief descriptions of these algorithms are provided below.

2.1 Genetic Algorithm

The Genetic Algorithms (GAs) were presented by Holland [31] that modulate the idea of the Evolutionary Algorithm, through addition of a phase referred to as crossover. GA is known to include a random number-generator, genetic operators for reproduction, and a fitness evaluation unit. The main steps of the GA algorithm are presented as follows:

1: Initialize Population,
2: repeat,
3: Evaluation,
4: Reproduce,
5: Crossover,
6: Mutation,
7: until requirements are met.

In the initialization step, a random solution (x_i) is generated and then, the value of cost function $f(x)$ for every chromosome in the population is evaluated.

In the reproduction step, two chromosomes from the population are selected and the chromosome with the higher fitness value has a better opportunity to be chosen. In the crossover step, two new off springs are generated by the crossover operator applied to parents chosen. In the mutation step, if a random number within the range (0, 1) is less than the mutation rate (*MP*), the parameter or parameters of the offspring are mutated to introduce diversity between parents and the offspring. Then, the parents are discarded, and the offspring are kept in the population.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) presented by Kennedy and Eberhart [32] models the collective behaviors of birds flocking, or fish schooling. In the algorithm, each particle uses its previous experience while setting its own position for the best position in the track. The main steps of the PSO algorithm are presented as follows:

1: Initialize Population,
2: repeat,
3: Evaluate,
4: Update the best experience of all particles,
5: Choose the best particle,
6: Calculate particles' velocities,
7: Update particles' positions,
8: until requirements are met.

In the initialization step, a random position (x_i) is generated for each particle using Eq. (3), and then, the fitness function $f(x)$of each particle is computed in the evaluation step. The position of each particle is updated using Eq. (4) in updated step.

$$x_{ij} = x_j^{\min} + rand\,(0, 1)\left(x_j^{\max} - x_j^{\min}\right) \tag{3}$$

where x_{ij} is the position of i^{th} particle, i = 1 swarm size, j = 1 dimension of the problem, and $x_j^{\max}$, $x_j^{\min}$ refer to the lower and upper bound respectively.

$$x_i^{t+1} = x_i^t + v_i^t \tag{4}$$

where x_i^t is the position of each particle at iteration t, and v_i^t is the velocity of each particle at iteration t.

Then, the particle with best fitness $f(x)$ value chosen, and velocity is updated using Eq. (5):

$$v_i^{t+1} = w \cdot v_i^t + c_1 \cdot r_1\left(pb_i - x_i^t\right) + c_2 \cdot r_2\left(gb_i - x_i^t\right) \tag{5}$$

where w is inertia weight, c_1 and c_2 refers to cognitive component and social component sequentially, pb is personal best position of i^{th} particle, gb is global best position of any particle and both r_1, r_2 indicates a random value within the range (0, 1).

2.3 Differential Evolution Algorithm (DE)

The differential evolution algorithm is an intelligent search algorithm proposed by Storn and Price [29]. In DE, all variables are represented as a real number and the crossover, mutation and selection operators are iterated in DE. The main steps of the DE algorithm are presented as follows:

1: Initialize Population,
2: Evaluation,
3: repeat,
4: Mutation,
5: Crossover,
6: Evaluation,
7: Selection,
8: until requirements are met in.

In the initialization step, a random individual (x_i) is generated using Eq. (3) and then, the cost function $f(x_i)$ of each individual is computed in the evaluation step.

In the mutation step, an x_i^{mutation} individual is generated using Eq. (6).

$$x_i^{\text{mutation}} = x_{r1} + F\left(x_{r3} - x_{r2}\right) \tag{6}$$

where r_1, r_2 and r_3 are random integers generated, each one is different from each other, and are all not equal to x_i^{mutation}, F is scaling factor that is generated randomly in the range of (0, 1).

In the crossover step, an (x_i) individual is mixed with the (x_i^{mutation}) individual in order to produce (x_i^{trial}) individual using Eq. (7).

$$x_i^{\text{trial}} = \begin{cases} x_i^{\text{mutation}} & \text{if} \quad j_{rand} \le CR \\ x_i & \text{others} \end{cases} \tag{7}$$

where j is a random integer number and CR is a crossover rate generated within the range of (0, 1).

In the selection step, each (x_i^{trial}) individual competes with (x_i) individual and the best one is saved in the population.

3 PROPOSED DE-BASED CRYPTANALYSIS

In this study, cryptanalysis steps can be achieved by considering the following pseudocode code:

- Initialize population *pop*(*i*) using Eq. (3), control parameters,
- while requirements are met do,
- for every *suggested key*(*i*) ∈ *pop*(*i*) do
  - Evaluate *f* (*suggested key*(*i*)) using Eq. (8),
  - Apply mutation operator to create *trial suggested key*(*i*) using Eq. (6),
  - Apply crossover operator to create offspring *suggested key**(*i*) using Eq. (7),
  - If *f* (*suggested key**(*i*)) ≤ *f* (*suggested key*(*i*)) then
    - Insert *suggested key**(*i*) to *pop*(*i* + 1),
  - else
    - Insert *suggested key*(*i*) to *pop*(*i* + 1),
  - end if
- end for
- end while

Table 1 English unigram frequencies (in percent %)

No	Unigram	Frequencies	No	Unigram	Frequencies
1	A	8.55	14	N	7.17
2	B	1.60	15	O	7.47
3	C	3.16	16	P	2.07
4	D	3.87	17	Q	0.10
5	E	12.10	18	R	6.33
6	F	2.18	19	S	6.73
7	G	2.09	20	T	8.94
8	H	4.96	21	U	2.68
9	I	7.33	22	V	1.06
10	J	0.22	23	W	1.83
11	K	0.81	24	X	0.19
12	L	4.21	25	Y	1.72
13	M	2.53	26	Z	0.11

The cost function (fitness function) has a critical role in the efficiency of a metaheuristic algorithm since the cost function discovers the integrity of the possible key. The objective here is to offer meaningful and comparable value to guide the algorithm. The solution with high fitness function has a chance to remain in the next generation and to continue towards optimal solutions. The fitness function provides local optimal solutions, and its quality is higher if a global optimal is achieved.

In the present study we employed a fitness function defined using the unigram and bigram statistics of the language considered [22, 25]. The fitness function f of a suggested key K can be defined by Eq. (8):

$$f(K) = \lambda_1 \sum_{i=1}^{\text{unigram}} \left| OFM(i) - EFM(i) \right| + \lambda_2 \sum_{i=1}^{\text{bigram}} \left| OFB(i) - EFB(i) \right| \tag{8}$$

where K is the key used to decode the message, $OFM(i)$ and $EFM(i)$ are the observed and expected frequencies for i^{th} monogram, respectively, $OFB(i)$ and $EFB(i)$ are the observed and expected frequencies for i^{th} bigram, respectively. λ_1 and λ_2 are the weights assigned to unigram and bigram statistics respectively. The optimal weights, $\lambda_1 = 0.23$ and $\lambda_2 = 0.77$ are found in [33] based on the percentages of the retrieved words to be correct at different lengths for both ciphertext and keyword.

Table 2 English bigram frequencies (in percent %)

No	Unigram	Frequencies	No	Unigram	Frequencies
1	TH	2.71	16	OR	1.06
2	HE	2.33	17	EA	1.00
3	IN	2.03	18	TI	0.99
4	ER	1.78	19	AR	0.98
5	AN	1.61	20	TE	0.98
6	RE	1.41	21	NG	0.89
7	ES	1.32	22	AL	0.88
8	ON	1.32	23	IT	0.88
9	ST	1.25	24	AS	0.87
10	NT	1.17	25	IS	0.86
11	EN	1.13	26	HA	0.83
12	AT	1.12	27	ET	0.76
13	ED	1.08	28	SE	0.73
14	ND	1.07	29	OU	0.72
15	TO	1.07	30	OF	0.71

Table 3 Turkish unigram frequencies (in percent %)

No	Unigram	Frequencies	No	Unigram	Frequencies
1	A	11.82	16	O	2.47
2	E	9.00	17	Ü	1.97
3	İ	8.34	18	Ş	1.83
4	N	7.29	19	Z	1.51
5	R	6.98	20	G	1.32
6	L	6.07	21	Ç	1.19
7	I	5.12	22	H	1.11
8	K	4.7	23	Ğ	1.07
9	D	4.63	24	V	1.00
10	M	3.71	25	C	0.97
11	Y	3.42	26	Ö	0.86
12	U	3.29	27	P	0.84
13	T	3.27	28	F	0.43
14	S	3.03	29	J	0.03
15	B	2.76			

In this study, we considered some texts written in the English and Turkish languages. The highest frequencies of unigrams and bigrams in these languages are found through computation. The frequency values observed for unigrams are subtracted from the normal frequencies and the sum of the differences is calculated. The same procedure is

performed for bigrams. Tab. 1 and Tab. 2 present the expected values for unigram and bigram [34] generated using around 4.5 billion characters in English, similarly, Tab. 3 and Tab. 4 present the expected values in Turkish [35].

Table 4 Turkish bigram frequencies (in percent %)

No	Unigram	Frequencies	No	Unigram	Frequencies
1	AR	0.02273	16	Dİ	0.01021
2	LA	0.02013	17	ND	0.00980
3	AN	0.01891	18	RA	0.00976
4	ER	0.01822	19	AL	0.00974
5	İN	0.01674	20	AK	0.00967
6	LE	0.01640	21	İL	0.00870
7	DE	0.01475	22	Rİ	0.00860
8	EN	0.01408	23	ME	0.00785
9	IN	0.01377	24	Lİ	0.00782
10	DA	0.01311	25	OR	0.00782
11	İR	0.01282	26	NE	0.00738
12	Bİ	0.01253	27	RI	0.00733
13	KA	0.01155	28	BA	0.00718
14	YA	0.01135	29	Nİ	0.00716
15	MA	0.01044	30	EL	0.00710

4 EXPERIMENTS

In the first part of the experiments, the results of the proposed DE algorithm on the cryptanalysis of Vigenere cipher are presented. In the second part, the results of the DE, GA and PSO algorithms on the cryptanalysis of Vigenere cipher are compared to examine the efficiency of the DE method over the GA and PSO algorithms.

In all experiments, we assume that both plaintext and ciphertext include only the English (26 Letters) and Turkish (29 Letters) alphabets. We investigated the keywords with five different sizes (5, 10, 15, 20 and 25), and plain texts with four different lengths (250, 500, 750 and 1000). The best values of the control parameters used in this experiment are obtained from grid search and presented in Tab. 5. Each experiment is repeated 30 times and statistics of these runs are reported in the results.

Table 5 Control parameters values of each algorithm

Control parameters	DE	GA	PSO
Population size (*PS*):	50	250	250
Crossover rate (*CR*):	0.2	0.75	
Mutation rate (*MP*):		0.75	
Social component (c_1):			2
Cognitive component (c_2):			1.75
Inertia weight (w):			0.7
Scaling Factor (F):	1		
Max generations:	600	120	120

4.1 Experiment 1: De Algorithm in the Vigenere Cipher Cryptanalysis

Tab. 6 and Tab. 7 display the retrieved key characters and the fitness levels obtained for English and Turkish ciphertexts using DE algorithm, respectively.

When the ciphertext size is less than 250 character, the minimum and maximum number of key characters recovered correctly is less than the minimum and maximum number of key characters recovered correctly in solving a ciphertext of size 500, 750, 1000. Likewise, the mean of the number of the key characters recovered correctly is less than the mean of the number of key characters recovered correctly in solving a cyphertext of size 500, 750, 1000. The standard deviation of the number of key characters recovered correctly with ciphertext of size 250 is higher than the standard deviation of the number of key characters recovered correctly with ciphertext of size 500, 750, 1000.

With an increase in ciphertext length, (> 250 character) the number of key characters recovered correctly increases as well because the reliability produced by higher size of ciphertext made the fitness higher and a good approximation to the expected values is obtained. From the results, the iteration cycle is directly related to the key and ciphertext length. When the ciphertext is getting small, the iteration cycle is increasing in the decryption. The encrypted text with more characters makes the key estimation more effective and reduces the iteration cycle needed. Interestingly, the accuracy to find the keys is typically higher in Turkish than that in English texts, even if the ciphertext is short (≤ 250 character) because the average length of the words in the Turkish language is 6.1 letters about 30% more than that of the English language, moreover, the short words (with 3 to 8 letters) represent over 60% of total usage in the Turkish language and that provides a wealth of information in cryptanalysis [35].

4.2 Experiment 2: Comparison of DE, GA and PSO in the Vigenere Cipher Analysis

In this part of the study, the proposed DEalgorithm is compared to other search algorithms including GA and PSO on Vigenere cipher analysis. The best, mean and standard deviations of the maximum number of key characters recovered correctly by DE, GA and PSO algorithms are considered to validate the results. These results are shown in Tab. 8 for English ciphertexts and in Tab. 9 for Turkish ciphertexts.

When the key size is equal or less than 5 characters, the best value is the same for all three algorithms although for all cases, DE has the highest mean value and the minimum standard deviation compared to GA and PSO. When the ciphertext size is greater than 250 characters, GA produced the best value and the highest mean value with minimum standard deviation compared to the PSO. Best values produced by both GA and DE are very close, but the mean value and standard deviation produced by DE is better. From the results, it is seen that GA algorithm retrieves almost all characters, when the size of the keywords is higher than 15 and the size of ciphertexts is higher than 250. PSO algorithm is efficient when Vigenere cipher uses a smaller length of key (less than 5characters) while it is not so efficient when dealing with longer key lengths (greater than 5 characters). It is also found that DE is more efficient than GA, when the ciphertext length is equal or less than 500 characters. Furthermore, the best mean and standard deviation values of the number of key characters recovered correctly produced by DE are better than those obtained from GA and PSO.

Table 6 Results of DE algorithm for different key sizes and different lengths of the ciphertexts in English, NKCRC: The number of key characters recovered correctly

Ciphertext length	Key Size	Minimum NKCRC	Maximum NKCRC	Mean of NKCRC	Standard deviation of NKCRC	Minimum fitness	Maximum fitness	Mean of fitness	standard deviation of fitness
250	5	0	5	4.6667	0.9589	61.0584	86.6142	82.6007	8.2841
	10	0	8	4.1000	3.4276	55.7505	72.3724	62.1395	5.5459
	15	0	10	2.0333	3.2851	55.7368	66.1111	58.1354	2.3933
	20	0	9	1.3667	2.4563	55.0379	61.3401	56.9359	1.4221
	25	0	5	1.2667	1.6386	56.5267	59.7120	57.8161	0.8186
500	5	5	5	5.0000	0.0000	205.8708	205.8708	205.8708	0.0000
	10	9	10	9.9333	0.2537	190.9625	205.8708	204.8769	3.7824
	15	14	15	14.9333	0.2537	197.4072	205.8708	205.5551	1.5498
	20	18	19	18.9667	0.1826	195.6625	205.9255	205.5551	1.8748
	25	23	24	23.9000	0.3051	206.3602	207.9439	207.7855	0.4832
750	5	5	5	5.0000	0.0000	342.4439	342.4439	342.4439	0.0000
	10	10	10	10.0000	0.0000	342.4439	342.4439	342.4439	0.0000
	15	15	15	15.0000	0.0000	342.4439	342.4439	342.4439	0.0000
	20	19	20	19.9667	0.1826	338.0419	342.4439	342.2972	0.8037
	25	24	25	24.9000	0.3051	334.7255	342.4439	341.9660	1.6042
1000	5	5	5	5.0000	0.0000	472.2756	472.2756	472.2756	0.0000
	10	10	10	10.0000	0.0000	472.2756	472.2756	472.2756	0.0000
	15	14	15	14.9667	0.1826	451.8462	472.2756	471.5946	3.7299
	20	18	20	19.9000	0.4026	440.0519	472.2756	470.6138	6.6081
	25	24	25	24.8667	0.3457	455.2919	472.2756	470.4147	4.9309

Table 7 Results of DE algorithm for different key sizes and different lengths of the ciphertexts in Turkish, NKCRC: The number of key characters recovered correctly

Ciphertext length	Key Size	Minimum NKCRC	Maximum NKCRC	Mean of NKCRC	Standard deviation of NKCRC	Minimum fitness	Maximum fitness	Mean of fitness	standard deviation of fitness
250	5	5	5	5.0000	0.0000	87.6893	87.6893	87.6893	0.0000
	10	8	10	9.7667	0.5683	82.5173	87.6893	87.4076	1.0862
	15	6	15	11.0000	2.4635	77.4866	88.4361	86.3848	2.9394
	20	2	16	9.6000	2.6987	64.4715	86.8992	74.5462	4.7858
	25	3	12	8.1000	2.2491	65.5436	78.6273	70.4414	2.9952
500	5	5	5	5.0000	0.0000	208.7812	208.7812	208.7812	0.0000
	10	10	10	10.0000	0.0000	208.7812	208.7812	208.7812	0.0000
	15	15	15	15.0000	0.0000	208.7812	208.7812	208.7812	0.0000
	20	19	20	19.9667	0.1826	204.6937	208.7812	208.6450	0.7463
	25	24	24	24.0000	0.0000	209.5443	209.5443	209.5443	0.0000
750	5	5	5	5.0000	0.0000	332.4877	332.4877	332.4877	0.0000
	10	10	10	10.0000	0.0000	332.4877	332.4877	332.4877	0.0000
	15	15	15	15.0000	0.0000	332.4877	332.4877	332.4877	0.0000
	20	19	20	19.9333	0.2537	320.2477	332.4877	331.7050	2.9814
	25	23	25	24.9333	0.3651	317.8508	332.4877	331.9998	2.6723
1000	5	4	5	4.9667	0.1826	393.5308	459.2877	457.0958	12.0055
	10	10	10	10.0000	0.0000	459.2877	459.2877	459.2877	0.0000
	15	15	15	15.0000	0.0000	459.2877	459.2877	459.2877	0.0000
	20	20	20	20.0000	0.0000	459.2877	459.2877	459.2877	0.0000
	25	24	25	24.9333	0.2537	444.4208	459.2877	458.2966	3.7719

Table 8 The correct key-based comparison for English letters

Ciphertext length	Key Size	Best value			Mean value			Standard deviation		
		GA	PSO	DE	GA	PSO	DE	GA	PSO	DE
250	5	5	5	5	3.2333	1.3667	4.6667	1.8323	1.5862	0.9589
	10	6	4	8	0.5333	0.5333	4.1000	1.5916	1.2521	3.4276
	15	11	4	10	0.9333	0.2333	2.0333	2.4059	0.7739	3.2851
	20	7	5	9	0.3333	0.6333	1.3667	1.2954	1.2452	2.4563
	25	3	5	5	0.5000	0.5667	1.2667	0.9738	1.1043	1.6386
500	5	5	5	5	4.9667	3.3000	5.0000	0.1826	1.0875	0.0000
	10	10	7	10	9.4333	5.1000	9.9333	0.6261	1.1552	0.2537
	15	15	11	15	13.1000	6.7333	14.9333	1.6049	2.0667	0.2537
	20	20	11	19	14.7333	6.3000	18.9667	2.0833	2.2614	0.1826
	25	23	13	24	16.9000	7.6333	23.9000	2.8929	3.0113	0.3051
750	5	5	5	5	5.0000	3.4333	5.0000	0.0000	0.8172	0.0000
	10	10	9	10	9.6667	5.5667	10.0000	0.5467	1.4547	0.0000
	15	15	12	15	13.4333	7.4000	15.0000	1.5013	2.0443	0.0000
	20	20	11	20	16.9333	7.6667	19.9667	1.5960	1.4933	0.1826
	25	25	12	25	20.4000	8.4333	24.9000	2.4439	2.4023	0.3051
1000	5	5	5	5	4.9667	3.5333	5.0000	0.1826	0.9732	0.0000
	10	10	8	10	9.7667	5.7333	10.0000	0.4302	1.1427	0.0000
	15	15	10	15	13.4667	7.1333	14.9667	1.4320	1.3578	0.1826
	20	20	11	20	17.6667	8.0000	19.9000	1.5610	1.9652	0.4026
	25	25	18	25	21.2000	9.9333	24.8667	2.0578	2.6901	0.3457

Table 9 The correct key-based comparison for Turkish letters

Ciphertext length	Key Size	Best value			Mean value			Standard deviation		
		GA	PSO	DE	GA	PSO	DE	GA	PSO	DE
250	5	5	5	5	4.6333	3.3333	5.0000	0.7184	0.9223	0.0000
	10	10	7	10	7.7667	3.6667	9.7667	2.3295	1.4223	0.5683
	15	13	8	15	7.8000	2.8667	11.0000	3.2632	2.1772	2.4635
	20	15	8	16	6.9000	3.1333	9.6000	3.5365	2.3004	2.6987
	25	15	9	12	8.0667	3.2333	8.1000	3.7040	1.7750	2.2491
500	5	5	5	5	5.0000	3.5667	5.0000	0.0000	0.6261	0.0000
	10	10	7	10	9.6333	5.1333	10.0000	0.5561	0.9732	0.0000
	15	15	9	15	13.9333	6.0000	15.0000	1.1427	1.2034	0.0000
	20	19	10	20	15.7000	6.4333	19.9667	2.4233	1.4782	0.1826
	25	24	12	24	18.9667	7.2000	24.0000	2.8945	2.1560	0.0000
750	5	5	5	5	5.0000	3.5333	5.0000	0.0000	0.6814	0.0000
	10	10	7	10	9.7333	5.2333	10.0000	0.5208	1.1943	0.0000
	15	15	8	15	13.9000	5.9333	15.0000	1.0619	1.3113	0.0000
	20	20	12	20	17.2667	7.3667	19.9333	1.8182	1.7317	0.2537
	25	23	12	25	19.1333	8.4333	24.9333	2.5695	2.0625	0.3651
1000	5	5	5	5	4.9667	3.8333	4.9667	0.1826	0.7915	0.1826
	10	10	8	10	9.6333	5.0667	10.0000	0.6149	1.0483	0.0000
	15	15	10	15	13.9000	6.5333	15.0000	1.1552	1.6344	0.0000
	20	20	10	20	17.2667	7.5333	20.0000	1.8182	1.2521	0.0000
	25	25	13	25	21.0667	8.5667	24.9333	2.0998	2.2234	0.2537

5 CONCLUSION

In the Vigenere cryptoanalysis, there is a huge range of possible keys, and the manual cryptanalysis and the statistical techniques are inefficient when the key length is longer. This study aimed to analyze the suitability of DE algorithm as a cryptanalytic tool. The results show that it is an efficient method for the Vigenere cipher. Consequently, DE algorithm has ability to retrieve all the characters of the keyword for the key size is 25 characters and ciphertext size is more than 250 characters, while GA and PSO algorithm can retrieve the entire key correctly when the length of keys is small. Based on experimental results, we can conclude that the results of DE are better than those obtained from GA and PSO in the cryptanalysis of Vigenere cipher. Also, computational results and comparisons demonstrate that iteration cycle is directly related to the key and ciphertext length, also the accuracy to find the keys is typically higher in Turkish texts than that in English texts, even if the ciphertext is short (≤ 250 character). Tailoring efficient fitness functions remains to be studied as our future work.

Acknowledgements

This study is supported by Erciyes University Research Projects Unit with grant number FDK-2016-7085.

6 REFERENCES

[1] Wesley, S, W. (2019). Cryptography and network security: Principles and practice. Hoboken, NJ: Pearson Education.

[2] Al-Kadit, I. A. (1992). Origins Of Cryptology: The Arab Contributions. *Cryptologia*, *16*(2), 97-126. https://doi.org/10.1080/0161-119291866801

[3] Azizi, A. (2019). Arabic Cryptography and Steganography in Morocco. *Codes, Cryptology and Information Security Lecture Notes in Computer Science*, 43-54. https://doi.org/10.1007/978-3-030-16458-4_4

[4] Kahn, D. (1997). *The codebreakers: The comprehensive history of secret communication from ancient times to the Internet.* New York: Scribners and Sons.

[5] Wigderson, A. (2019). *Mathematics and computation*. New Jersey: Princenton University Press.

[6] Simmons, G. J. (1979). SymmetricandAsymmetric Encryption. *ACM Computing Surveys*, *11*(4), 305-330. https://doi.org/10.1145/356789.356793

[7] Poh, G. S., Chin, J., Yau, W., Choo, K. R., & Mohamad, M. S. (2017). Searchable Symmetric Encryption. *ACM Computing Surveys*, *50*(3), 1-37. https://doi.org/10.1145/3064005

[8] Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice, Global Edition*. Harlow, United Kingdom: Pearson.

[9] Holden, J. (2018). *Themathematics of secrets: Cryptography from caesar ciphers to digital encryption*. Princeton, NJ: Princeton University Press. https://doi.org/10.1515/9780691184555

[10] Padhye, S., Sahu, R. A., & Saraswat, V. (2018). *Introduction to cryptography*. Boca Raton, FL: CRC Press, Taylor & Francis Group. https://doi.org/10.1201/9781315114590

[11] Spillman, R., Janssen, M., Nelson, B., & Kepner, M. (1993). Use Of A Genetic Algorithm In The Cryptanalysis Of Simple Substitution Ciphers. *Cryptologia*, *17*(1), 31-44. https://doi.org/10.1080/0161-119391867746

[12] Matthews, R. A. (1993). The Use Of Genetic Algorithms In Cryptanalysis. *Cryptologia*, *17*(2), 187-201. https://doi.org/10.1080/0161-119391867863

[13] Clark, A. (1994). Modern optimisation algorithms for cryptanalysis. *In Proceedings of ANZIIS '94 - Australian New Zealnd Intelligent Information Systems Conference*. Australia,: IEEE. https://doi.org/10.1109/ANZIIS.1994.396969

[14] Clark, A., Dawson, E., & Nieuwland, H. (1996). Cryptanalysis of polyalphabetic substitution ciphers using a parallel genetic algorithm. *In Proceedings of International Symposium on Information and its Applications* ,17-20.

[15] Clark, A. & Dawson, E. (1997). A Parallel Genetic Algorithm For Cryptanalysis Of The Polyalphabetic Substitution Cipher. *Cryptologia*, *21*(2), 129-138. https://doi.org/10.1080/0161-119791885850

[16] Clark, A. & Dawson, E. (1998). Optimisation heuristics for the automated cryptanalysis of classical ciphers. *Journal of Combinatorial Mathematics and Combinatorial Computing*, *28*, 63-86.

[17] Dimovski, A. & Gligoroski, D. (2003). Attacks on the transposition ciphers using optimization heuristics. *Proceedings of ICEST*, 1-4.

[18] Verma, A. K., Dave, M., & Joshi, R. C. (2007). Genetic Algorithm and Tabu Search Attack on the Mono-Alphabetic

Substitution Cipher in Adhoc Networks. *Journalof Computer Science*, *3*(3), 134-137. https://doi.org/10.3844/jcssp.2007.134.137

[19] Song, J., Yang, F., Wang, M., & Zhang, H. (2008). Cryptanalysis of Transposition Cipher Using Simulated Annealing Genetic Algorithm. *Advances in Computationand Intelligence Lecture Notes in Computer Science*, 795-802. https://doi.org/10.1007/978-3-540-92137-0_87

[20] Garg, P. (2009). Genetic Algorithms, Tabu Search and Simulated Annealing: A Comparison Between Three Approaches for The Cryptanalysis of Transposition Cipher. *Journal of Theoretical & Applied Information Technology*, *5*(4).

[21] Omran, S. S., Al-Khalid, A. S., & Al-Saady, D. M. (2011). A cryptanalytic attack on Vigenère cipher using genetic algorithm. *2011 IEEE Conference on Open Systems*. https://doi.org/10.1109/ICOS.2011.6079312

[22] Bhateja, A. & Kumar, S. (2014). Genetic Algorithm with elitism for cryptanalysis of Vigenere cipher. *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT).* https://doi.org/10.1109/ICICICT.2014.6781311

[23] Boryczka, U. & Dworak, K. (2014). Genetic Transformation Techniques in Cryptanalysis. Intelligent Information and Database Systems Lecture Notes in Computer Science, 147-156. https://doi.org/10.1007/978-3-319-05458-2_16

[24] Uddin, M. & Youssef, A. (2006). An Artificial Life Technique for the Cryptanalysis of Simple Substitution Ciphers. *2006 Canadian Conference on Electrical and Computer Engineering*. https://doi.org/10.1109/CCECE.2006.277769

[25] Bhateja, A. K., Bhateja, A., Chaudhury, S., & Saxena, P. (2015). Cryptanalysis of Vigenere cipher using Cuckoo Search. *Applied Soft Computing*, *26*, 315-324. https://doi.org/10.1016/j.asoc.2014.10.004

[26] Luthra, J. & Pal, S. K. (2011). A hybrid Firefly Algorithm using genetic operators for the cryptanalysis of a monoalphabetic substitution cipher. *2011 World Congress on Information and Communication Technologies*. https://doi.org/10.1109/WICT.2011.6141244

[27] Sabonchi, A. K. & Akay, B. (2017). Cryptanalytic of Substitution Ciphers by Artificial Bee Colony Algorithm Guided by Statistics based Fitness Function. *In The International Advanced Technologies Symposium, IATS17*, Elaziğ, Türkiye, 3874-3879.

[28] Sabonchi, A. K., & Akay, B. (2017). Cryptanalysis using Artificial Bee Colony Algorithm Guided by Frequency based Fitness Value. In 1st International Symposium on Multidisciplinary Studies and Innovative Technologies, ISMSIT(pp. 334-338). Tokat, Türkiye.

[29] Storn, R. & Price, K. (1997). Differential Evolution-A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341-359. https://doi.org/10.1023/A:1008202821328

[30] Das, S., Mullick, S. S., & Suganthan, P. (2016). Recent advances in differential evolution - An updated survey. *Swarm and Evolutionary Computation, 27*, 1-30. https://doi.org/10.1016/j.swevo.2016.01.004

[31] Holland, J. H. (1975). Adaptation in natural and artificial systems: An introductory analysis with application to biology. Control and Artificial Intelligence. https://doi.org/10.7551/mitpress/1090.001.0001

[32] Kennedy, J. & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*. Australia, Perth, WA: IEEE. https://doi.org/10.1109/ICNN.1995.488968

[33] Dureha, A. & Kaur, A. (2013). A Generic Genetic Algorithm to Automate an Attack on Classical Ciphers. *International Journal of Computer Applications*, *64*(12), 20-25. https://doi.org/10.5120/10687-5588

[34] James, I. (2012). Practical cryptography. Retrieved November 30, 2017, from http://practicalcryptography.com/ciphers/

[35] Dalkılıç, M. E. & Dalkılıç, G. (2002). On the Cryptographic Patterns and Frequencies in Turkish Language. *In International Conference on Advances in Information Systems*, *2457*, ADVIS 2002. Lecture Notes in Computer Science, pp. 144-153). Berlin: Springer. https://doi.org/10.1007/3-540-36077-8_14

Contact information:

Arkan Kh Shakr SABONCHI, PhD Student,
(Corresponding author)
Erciyes University,
Computer Engineering Department,
38039, Melikgazi, Kayseri, Turkey
E-mail**:** arkankhaleel@gmail.com

Bahriye AKAY, PhD, Associate Professor,
Erciyes University,
Computer EngineeringDepartment,
38039, Melikgazi, Kayseri, Turkey
E-mail: bahriye@erciyes.edu.tr