ISSN 1330-3651 (Print), ISSN 1848-6339 (Online) https://doi.org/10.17559/TV-20190422225110
Original scientific paper

A Binomial Crossover Based Artificial Bee Colony Algorithm for Cryptanalysis of Polyalphabetic Cipher

Arkan Kh Shakr SABONCHI*, Bahriye AKAY

Abstract: Cryptography is one of the common approaches to secure private data and cryptanalysis involves breaking down a coded cipher text without having the key. Cryptanalysis by brute force cannot be accepted as an effective approach and hence, metaheuristic algorithms performing systematic search can be applied to derive the optimal key. In this study, our aim is to examine the overall suitability of Artificial Bee Colony algorithm in the cryptanalysis of polyalphabetic cipher. For this purpose, using a number of different key lengths in both English and Turkish languages, basic Artificial Bee Colony algorithm (ABC) is applied in the cryptanalysis of Vigenere cipher. In order to improve the ABC algorithm's convergence speed, a modified binomial crossover based Artificial Bee Colony algorithm (BCABC) is proposed by introducing a binomial crossoverbased phase after employed bee phase for a precise search of global optimal solution. Different keys in various sizes, various cipher texts in both English and Turkish languages are used in the experiments. It is shown that optimal cryptanalysis keys produced by BCABC are notably competitive and better than those produced by basic ABC for Vigenere cipher analysis.

Keywords: Artificial Bee Colony; Binomial Crossover; Cryptanalysis; Polyalphabetic Cipher; Vigenere cipher

1 INTRODUCTION

Secure communication ensures unauthorised individuals not to be able to gain access to private data. This security is delivered through cryptography which generates an encrypted form of a given text with a key and facilitates confidentiality, message availability and integrity. Cryptanalysis deals with retrieving plain text and/or key from a cipher text, without the permission of the communicating parties or knowledge on the key [1].

Classical ciphers are categorized into substitution ciphers in which letters undergo systematic replacement across the course of the message for other letter and transposition ciphers in which the original text is broken down into blocks with a previously outlined size, depending on the permutation. Classical ciphers remain valuable and important owing to the fact that the majority of the widely utilised modern ciphers make use of classical ciphers in their basis. In actuality, the majority of the complex algorithms are created through combining transposition and substitution ciphers [2, 3]. Depending on the key, a cryptosystem may be divided into two systems: symmetric and asymmetric cryptosystems [2, 3]. Symmetric cryptosystems may be divided into two subgroups as monoalphabetic in which each character is substituted for another with a fixed substitution of another alphabet letter [2] and polyalphabetic cryptosystems in which an alphabet letter can be replaced without a fixed structure to a different letter depending on its plaintext position [3].The possible set of keys being the set of all potential alphabet permutations in the space of 26 letters by Monoalphabetic Substitution Ciphers equals more than 403 septillions which takes 12 trillion years to check all potential keys at a rate of a million keys per second [4]. Vigenere Cipher is a polyalphabetic cipher which encodes the text by replacing a plain-text character with another letter with the use of alphabet rows in a table, which are changed in line with the code word letters' indices [5]. In message encryption, the plain text and the key are formulated as integer sequences and divided into groups depending on key length. Let $P = (P_1, P_2, P_3, ..., P_n)$ a plain text block, $K = (K_1, K_2, K_3, ..., K_n)$ key; and cipher text block $C = (C_1, C_2, C_3, ..., C_n)$. Eq. (1) is used for encryption and Eq. (2) is used for decryption.

$$C_i = (P_i + K_i) \; mod \; 26 \tag{1}$$

$$P_i = (C_i - K_i) \; mod \; 26 \tag{2}$$

In Vigenere cipher, with regard to a key of size m, the key space is in the size of 26^m for English alphabet [6]. In this case, cryptanalysis by brute force is not practical due to its computational cost and stochastic optimization methods performing a systematic search based on randomization and previous experience are directed towards identifying the optimal key of classical cipher. In this study, one of the successful optimization algorithms in numerical optimization, Artificial Bee Colony algorithm (ABC) [7] was applied in cryptanalysis of polyalphabetic Vigenere cipher. The ABC algorithm in the class of swarm intelligence simulates the foraging behaviour of honey bees. It gained a success on many problems in various research fields [8]. ABC algorithm is a powerful search algorithm which makes it a potentially good candidate in finding cipher key. ABC algorithm has been applied to substitution cipher successfully [9, 10].This encourages further work into ABC application in Vigenere cipher cryptanalysis and this is the first time ABC algorithm is applied to cryptanalysis problem. In order to enhance convergence speed, ABC algorithm was modified by employing a binomial crossover phase between employed bee and onlooker bee phases. The binomial crossover produces the trial elements randomly by taking elements either from the mutation vector or from the exist elements, as described in Eq. (3).

$$B_i^j = \begin{cases} x_i^j & \text{if } R_j < CR \text{ or } j = K \\ y_i^j & otherwise \end{cases} \tag{3}$$

where B refers to the mixing results of the exist element, x_i, and that produced by mutation, y_i, R_j refers to a random value produced randomly for each j in range of (0, 1), CR

crossover rate which is randomly selected in range of (0, 1), and k is a randomly chosen from {1, …, n}, to include the whole elements from the mutation vector. The proposed approach is referred as Binomial Crossover based Artificial Bee Colony algorithm (BCABC) which integrates the advantage of ABC in exploration phase and a crossover operator in exploiting the information of neighbours. In the first part of this study, the parameter sensitivities of the basic ABC algorithm and BCABC algorithm are investigated and some values for the control parameters are recommended. By the best parameter configurations, the results produced by two algorithms are presented and compared based on statistical tools. The paper is organized as follows: In Section 2, related works on cryptanalysis using stochastic algorithms are presented, in Section 3, a brief description of ABC algorithm is explained, and the details of the proposed algorithm are given in Section 4. In Section 5, the experiments are explained, and the results are discussed. Finally, the last section is dedicated to the conclusion.

2 RELATED WORK

Some nature-inspired stochastic algorithms have been utilised in the cryptanalysis of classical cryptosystems. Spillman et al. [11] implemented a Genetic Algorithm (GA) to attack a Monoalphabetic Substitution Cipher. Furthermore, a GA system was adopted by Matthews [12], known as GENALYST, to break the transposition cipher. Clark [13] applied GA, Tabu Search (TS) and Simulated Annealing (SA) to cryptanalysis of substitution cipher. Moreover, Clark et al. [14] applied GA to attack on a polyalphabetic substitution cipher. Clark and Dawson [15] enhanced the work by integrating parallelism. Moreover, Clark and Dawson [16] presented a comparison among SA, GA and TS on the cryptanalysis of simple substitution ciphers. As a result, Dimovski and Gligoroski [17] highlighted three optimisation heuristics that could be applied in order to achieve transposition cipher breaking, namely SA, GA and TS. In the study of Verma et al. [18], a mono-alphabetic substitution cipher based on GA and TS for cryptanalysis was presented and subsequently compared the overall efficiency of such algorithms. An automated approach based on GA, TS and SA for the cryptanalysis of transposition cipher was developed by Song et al. [19] and Garg [20]. In addition, Omran et al. [21] developed a GA with the aim of attacking the Vigenere Cipher. Moreover, Bhateja and Kumar [22] devised an approach for the cryptanalysis of Vigenere cipher through GAs adopting elitism with a novel fitness function. In this regard, Boryczka and Dworak [23] considered the way in which evolutionary algorithms, including GAs, may be directed towards increasing the speed with which the cryptanalysis process of the transposition cipher can be increased. Further, Boryczka and Dworak [24] discussed the way in which evolutionary algorithms, including GAs, may be able to achieve the optimisation of the more complex cryptanalysis function. In addition, Uddin & Youssef [25] applied ACO in order to attack simple substitution ciphers. In the work carried out by Bhateja et al. [26], the suitability of the Cuckoo search algorithm (CS) in the cryptanalysis of the Vigenere cipher was investigated, whilst Luthra and Pal [27] directed their efforts towards examining the integration of mutation and crossover with the FA for cryptanalysis of the Mono-alphabetic cipher. In the study of Li et al. [28] which focused on presenting a hybrid of TS and GA, the aim was to break classical and modern ciphers, with the findings emphasising that the hybrid algorithms were most influential in regard to the transposition cipher, with a minimal effect witnessed on the hill cipher, and the Advanced Encryption Standard (AES) cipher demonstrating the lowest level of effect across all three algorithms. Ali and Mahmod [29] introduce a hybrid technique of Bees Algorithm (BA) SA for the cryptanalysis of simple substitution cipher. In the work which is carried out by Din et al. [30], they introduced some of the theoretical parts of stochastic algorithms and provided detailed cryptanalysis problems such as Stream ciphers and Block ciphers. In addition, Civicioglu and Besdok [31] analyzed statistically the numerical optimization problem successes of the CS, PSO, DE and ABC algorithms. Most of the techniques in the literature are unable to produce satisfactory results in the analysis of the Vigenere cipher when the key size exceeds 15 characters.

3 ARTIFICIAL BEE COLONY ALGORITHM

The Artificial Bee Colony algorithm defined by Karaboga [7] simulates the foraging behaviour of honey bees. In a bee colony, there are three different groups of bees, namely employed, onlooker and scout. Each employed bee exploits the food source in her memory and returns to the hive where their information is shared through communicative dancing whose frequency is related to the food source quality. The onlooker bees observe the dances, gain information, and accordingly decide which food source to fly and exploit. This type of communication allows more onlooker bees to be attracted by high quality sources. Importantly, if the nectar of a source is exhausted, the source is then abandoned and its employed bee becomes a scout. The scouts are not given any guidance when seeking out food; rather, they complete their own searches, exploring new areas to find new sources.

The ABC algorithm encompasses a number of key stages, highlighted as follows:

Initialization of control parameters, the number of food sources (SN), abandonment limit parameter (L), maxCycle:maximum Cycle Number, problem specific values: number of decision variables (D),decision variables (lower ($x_j^{\min}$) and upper ($x_j^{\max}$) bound).

****Initialization Phase****

1. The food source population (x_{ij}) is initialized by Eq. (4) and evaluated $fi(x_{ij})$.

$$x_{ij} = x_j^{\min} + rand(0, 1)(x_j^{\max} - x_j^{\min}) \quad (4)$$

where $i = 1, 2, ..., SN$, $j = 1, 2, ..., D$.

2. cycle = 1
3. **repeat**

****Employed Bee Phase****

4. **for** $i = 1$ to SN **do**

Produce a new solution ($x_{ij}^{\text{newsource}}$) by Eq. (5) and evaluate $fi(x_{ij})$:

$$x_{ij}^{\text{newsource}} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \quad (5)$$

where $i = 1, 2, ..., SN$, $j = 1, 2, ..., D$, $k \neq i$, and randomly chosen from $\{1, 2, ..., SN\}$, and $\emptyset_{ij}$ is a random real number in the range of $(-1, 1)$.

if $fi(x_{ij}^{\text{newsource}}) > fi(x_{ij})$, **then**, $(x_{ij}) = (x_{ij}^{\text{newsource}})$, counter $(i) = 0$

5. **else** counter (i) = counter (i) + 1; **end if; end for**
6. The probability values for solutions are calculated and determined by Eq. (6).

$$p_i = fitness_i / \sum_{i=1}^{foodsource} fitness_i \quad (6)$$

where fitnessi represents the food source fitness values.

****Onlooker Bee Phase****

7. **for** $t = 1$ to SN **do**
8. **if** $U(0, 1) < p_i$, **then** a new solution ($x_{ij}^{\text{newsource}}$) is produced by Eq. (5) and evaluated $fi(x_{ij}^{\text{newsource}})$.
9. **if** $fi(x_{ij}^{\text{newsource}}) > fi(x_{ij})$, **then**, $(x_{ij}) = (x_{ij}^{\text{newsource}})$, counter $(i) = 0$
10. **else** counter (i) = counter (i) + 1; **end if; end if; end for**

****Scout Bee Phase****

11. **if** counter (i) of the abandoned solution > L, **then** Replaced the abandoned solution with a new random solution (x_{ij}) generated by Eq. (4) and evaluated $fi(x_{ij})$, counter $(i) = 0$; **end if**
12. **cycle = cycle + 1**
13. **Until (maxCycle is reached).**

4 BINOMIAL CROSSOVER BASED COMBINATORIAL ARTIFICIAL BEE COLONY ALGORITHM (BCABC)

In the collective knowledge leading to swarm intelligence, the most critical aspect is social learning. In the case of the ABC algorithm, this is predominantly achieved by the unsupervised interaction among bees in the employed and the onlooker bees phases. In basic ABC algorithm, an information exchange occurs in each local search by changing one randomly chosen dimension based on the information coming from a random neighbour. In some cases, when the information exchange remains limited to one dimension, it may slow down the algorithm's convergence. In the proposed model, we integrated a binomial crossover phase after employed bee phase is completed to enhance local search ability of basic ABC algorithm. There are some studies in the literature in which a binomial crossover was used: Zaharie [32] analyzed a theoretical and a numerical point of the crossover variant on the behavior of the DE. In addition, Weber and Neri [33] compared the binomial crossover used in the DE with a variant binomial crossover, the results showed that this variant of the binomial crossover increases significantly the execution speed of the DE, especially in higher dimension problems. In the work carried out by Islam et al. [34], they proposed a new mutation strategy for the binomial crossover of the DE. Moreover, Lin et al. [35] presented theoretical analysis and comparative study for two new crossover methods, binomial crossover and exponential crossover.

Moreover, there are some studies in the literature in which ABC algorithm was integrated with crossover operators [36-40]. Some of them used crossover operator placed in the employed bee, onlooker bee or scout bee phase. Others used the crossover operator, with other improvements in the structure of the original ABC. Others used it with mutation operator as a separate phase. And others used real coded crossover operator as a separate phase after employed bee. Unlike previous studies, this study is characterized by performing binomial crossover as a separate phase between employed and onlooker bee phases. The binomial crossover is given below:

The pseudo code of Binomial crossover.

- Input crossover rate (***CR***);
- Select two parent in same length (***x, y***);
- **for** $i = 1$ to D **do**
  - **if** $U(0,1) \leq CR$ **then**
    - The new individual $(i) = y(i)$;
  - **else**
    - The new individual $(i) = x(i)$;
  - **end**
- **end**

As shown in Fig. 1, two parents X_i and Y_i are selected, then a random uniform number U is created in the range of (0, 1). If ($U > CR$) "1" that means successful experiment, but "0" means unsuccessful experiment ($U \leq CR$). The child C is taking an elements C, M, U and T from X_i, and I, O, A and L from Y_i.

X_i	C	O	M	P	U	T	E	R
Y_i	B	I	N	O	M	I	A	L
U_i	0	1	0	1	0	0	1	1
C_i	C	I	M	O	U	T	A	L

Figure 1 Binomial crossover

Main steps of BCABC algorithm are given below:

Values for control parameters are set and initial population is generated by Eq. (4).

Repeat (for all food source/solutions)

a. Employed Bee Phase
b. Binomial Crossover Phase
c. Onlooker Bee Phase
d. Scout Bee Phase

Until (the satisfaction of the requirements is completed).

5 EXPERIMENTS

In order to adapt the algorithms to cryptanalysis of Vigenere Cipher, first cipher text, key size and relative character frequencies are given to the algorithm. The algorithm generates alternative decryption keys which maximizes the cost function defined based on the relative frequencies specific to the language used in texts. Each food source which is assumed to be a key is initialized by

random selection by Eq. (4), with a random uniform sample from integers 0 (lower bound) to 25 (upper bound) in case English language is used. In order to establish viability of the key, each solution is evaluated in the cost function defined using frequency analysis in which the aim is making a comparison between frequencies in the decrypted text with those identified in target literature of the text (English, Turkish, etc.). The difference between the most frequent number of bigram and unigram of the decrypted text and that of the corresponding bigram and unigram frequencies in normal language used (English or Turkish) is used as fitness function. In the present work, a fitness function in Eq. (7) [22, 26] has been applied which depends on the unigram and bigram statistics of the language used.

$$f(K) = \lambda_1 \cdot \sum_{i=1}^{unigram} |OFM(i) - EFM(i)| + \lambda_2 \sum_{i=1}^{bigram} |OFB(i) - EFB(i)| \quad (7)$$

where K is the key applied, $OFM(i)$ and $EFM(i)$ are identified and the expected frequencies of each unigrams while $OFB(i)$ and $EFB(i)$ are identified and expected frequencies of each bigram. λ_1 and λ_2 are weights assigned to unigram and bigram statistics, respectively. We adopted the values $\lambda_1 = 0.23$ and $\lambda_2 = 0.77$ suggested as the best values in [41].

The expected values of unigram and bigrams in English were derived from approximately 4.5 billion characters of English text [42] and given in Appendix Tab. A1 to Tab. A2. Similarly, those in Turkish were taken from the work of [43] and given in Appendix Tab. A3 to Tab. A4. In the case of unigram, all 26 letters of the English alphabet and 29 letters in Turkish alphabet were considered. Regarding the bigrams, the most frequent 30 bigrams for both English and Turkish literature were utilised. In the first part of the experiments, various values are set for the control parameters of ABC and BCABC and the suitable parameter values yielding the best values are determined to be used in the next part of the study. In the second part of the experiments, the results of the algorithms on the cryptanalysis of Vigenere cipher are presented, ABC and BCABC algorithms are compared to analyse the efficiency of the proposed method over the basic ABC algorithm based on a statistical analysis. In all experiments, key sizes in {5, 10, 15, 20, 25} and varying sizes of Vigenere cipher text in {250, 500, 750, 1000} from both English and Turkish plain text files are employed.

In this study, as hardware 48 number Acer-Veritton Processor is used: Intel Core i7-2600K CPU @3.40 GHZ × 4 RAM: 3.8 GB Memory, Display Card: NVIDIA Corporation GF (GeForce GT 430), Disk Space: 500 GB and Windows 7 Professional, and as software MATLAB 2014 program is used.

5.1 Experiment 1: Parameter Tuning of the Metaheuristics

In the first part of the experiments, sensitivity of the algorithms against the values of control parameters is investigated. The same maximum evaluation number (30000) is utilised by ABC and BCABC algorithms as termination criterion. The search space is bounded with the range (0, 25) for English text and (0, 28) for Turkish text. The values {10, 20, 30, 40, 50, 100, 150, 200, 250} are analysed for the number of food sources (SN), key sizes (D) {5, 10, 15, 20, 25} and Vigenere cipher text sizes {250, 500, 750, 1000} each from the English and the Turkish. Therefore, 9 × 5 × 4 = 180 different test cases for each algorithm (ABC and BCABC) are used to determine the best population size. For each of the test cases, each experiment is repeated 30 times with different random seeds. Moreover, ABC and BCABC algorithms, experiments are repeated for five different values ($SN \times D \times 0.1$, $SN \times D \times 0.25$, $SN \times D \times 0.50$, $SN \times D \times 0.75$, $SN \times D \times 1$) of the parameter limit (L), with 5 × 4 × 5 = 100 different configurations tested and for each algorithm (ABCand BCABC). Each different case is repeated 30 times with different random seeds. In addition, experiments are repeated for three different values (0.1, 0.2, 0.5) of the parameter crossover rate (CR) hence related to BCABC algorithm, 3 × 4 × 5 = 60 different configurations tested, and each different case is repeated 30 times with different random seeds. The parameter values tested in Experiment 1 are listed in Tab. 1.

The best and mean values of the fitness values for various population values, different cipher sizes and various cipher text lengths are shown in Fig. 2 and Fig. 3, respectively. From the Figures, ABC algorithm produces better fitness values within (50 - 100) values of the population size while BCABC algorithm does that with a population size of (150 - 200). Fig. 4 and Fig. 5 show the best and mean fitness values obtained with different limit parameters, different cipher sizes and various cipher text lengths. It is seen that good results are obtained when the limit parameter is within ($0.5 \times SN \times D - 1 \times SN \times D$) for both ABC and BCABC algorithms. Within these ranges, we observe that the best results are obtained with the control parameter values in Tab. 2. Fig. 6 and Fig. 7 show the best and mean fitness values obtained with different CR parameters, different cipher sizes and various cipher text lengths. It is seen that good results are obtained when the CR is 0.2 for BCABC algorithms. Within these ranges, we observe that the best results are obtained with the control parameter values in Tab. 2.

Table 1 The parameter ranges used in Experiment 1

Control arameters	ABC	BCABC
Food Sources (SN)	10, 20, 30, 40, 50, 100, 150, 200, 250	10, 20, 30, 40, 50, 100, 150, 200, 250
Limit: (L)	SN × D × 0.1, SN × D × 0.25, SN × D × 0.50, N × D × 0.75, SN × D × 1	SN × D × 0.1, SN × D × 0.25, N × D × 0.50, SN × D × 0.75, SN × D × 1
Crossover rate (CR)		0.1, 0.2, 0.5

Table 2 The recommended values for the control parameters of each algorithm

Control parameters	ABC	BCABC
Food Sources (SN)	100	200
Limit (L)	0.75	0.1
Max generations:	300	75
CR		0.2

5.2 Experiment 2: Comparison of ABC and BCABC Algorithms

In the second part of the experiments, the results of the algorithms run with the control parameter values in Tab. 2 for the key sizes (D) {5, 10, 15, 20, 25} and Vigenere cipher text sizes {250, 500, 750, 1000} each from the English and Turkish are extracted and compared to each

other. Tab. 3 and Tab. 4 present the results of ABC algorithm in case English and Turkish alphabets, respectively and Tab. 5 and Tab. 6 present the results of BCABC algorithm for English and Turkish alphabets, respectively. from Tab. 3 to Tab. 6 reports the statistics of the number of key letters recovered correctly and fitness value statistics according to different cipher text lengths and key sizes. When the cipher text size equals 250 characters, the minimum and maximum number of key letters recovered correctly, is less than the minimum and maximum number of key letters recovered correctly in solving a cipher text of size 500, 750, 1000. As the cipher text length gets longer, the number of correct key letters increases as well because the approximation to expected values is more precise and the overall reliability in fitness is higher. Hence, when the cipher text is short ($\leq$ 250 character) or key length is high, it requires a higher number of cycles. Besides, the results obtained using Turkish alphabet are better than those obtained using English alphabet when the cipher text length was small. Tab. 7 provides a comparison of ABC and BCABC algorithm based on the statistics of the number of key letters recovered correctly while Tab. 8 gives a comparison of the algorithms based on the statistics of fitness values for both English and Turkish cases.

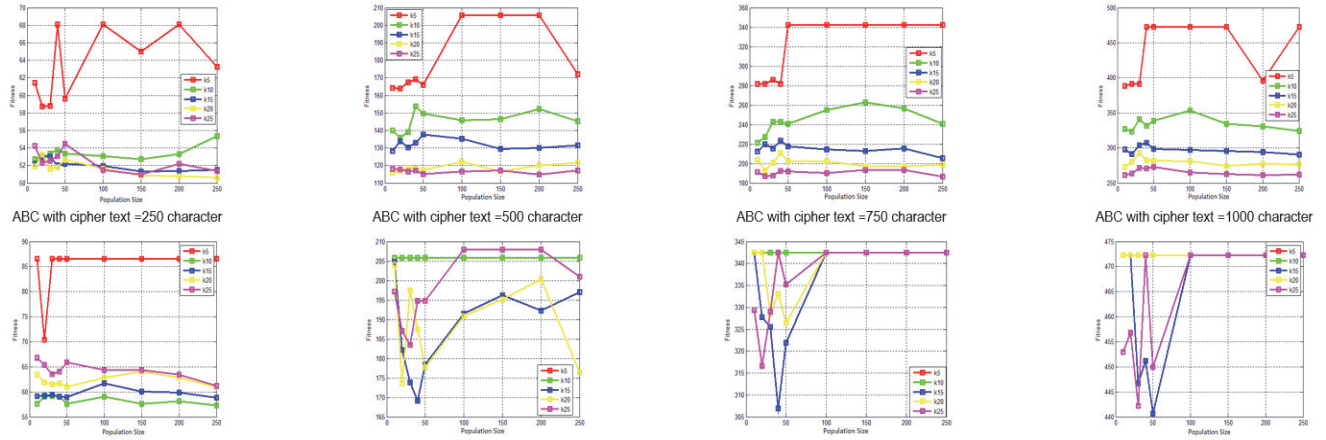


Figure 2 The best values of the fitness values for different *SN* values, various key sizes and cipher text sizes using ABC and BCABC algorithms

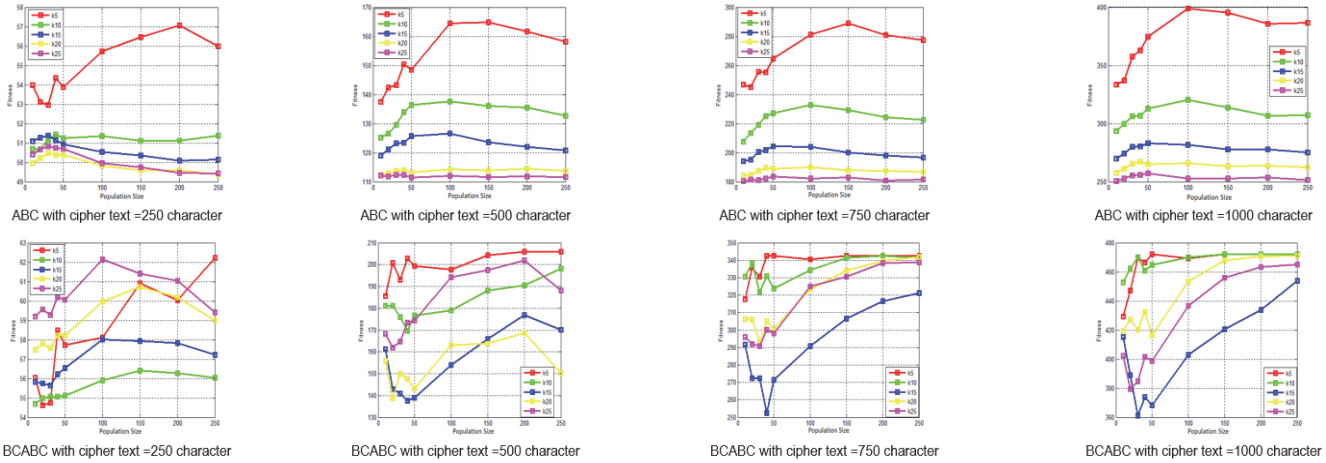


Figure 3 The mean values of the fitness values for different *SN* values, various key sizes and cipher text sizes using ABC and BCABC algorithms

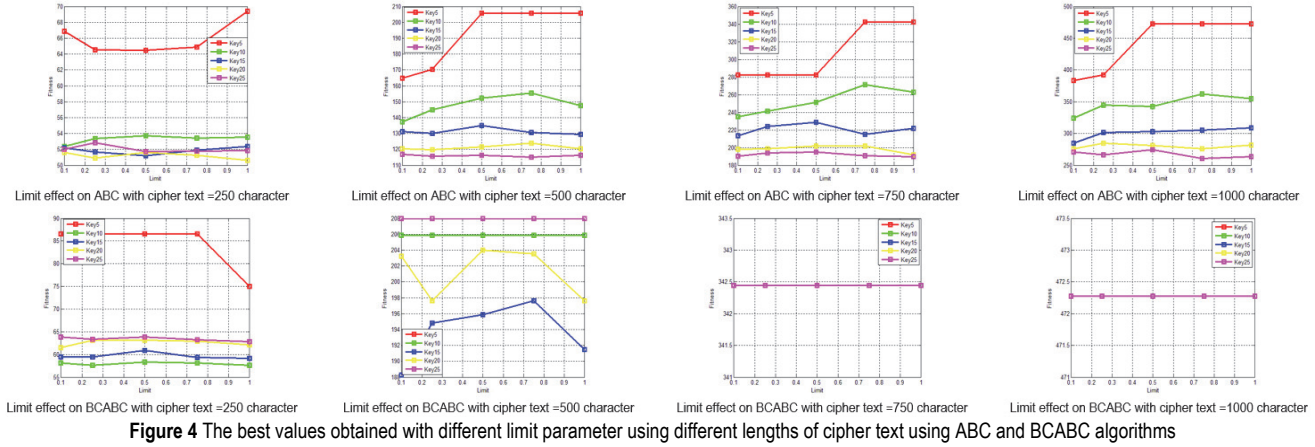
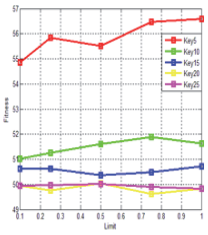
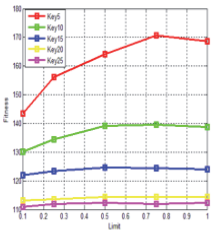
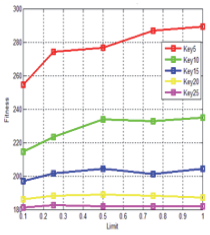
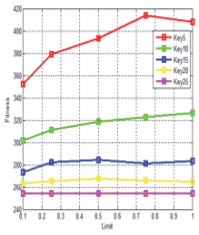


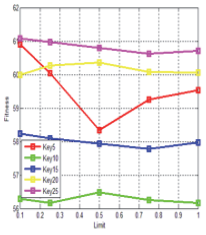
Figure 4 The best values obtained with different limit parameter using different lengths of cipher text using ABC and BCABC algorithms

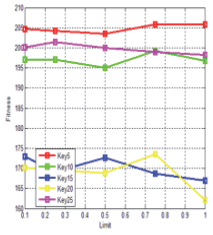

Limit effect on ABC with cipher text =250 character

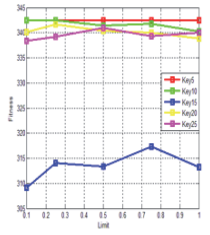

Limit effect on ABC with cipher text =500 character


Limit effect on ABC with cipher text =750 character


Limit effect on ABC with cipher text =1000 character


Limit effect on BCABC with cipher text =250 character


Limit effect on BCABC with cipher text =500 character


Limit effect on BCABC with cipher text =750 character

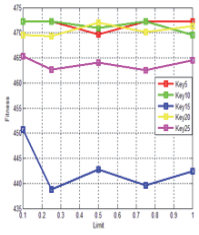
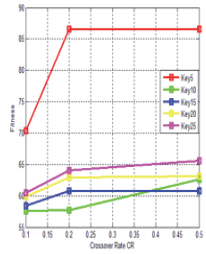
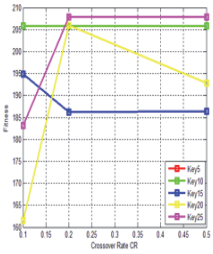
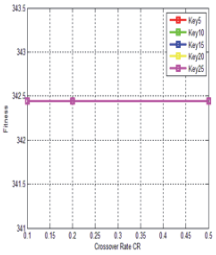

Limit effect on BCABC with cipher text =1000 character

Figure 5 The mean values obtained with different limit parameter using different lengths of cipher text using ABC and BCABC algorithms


CR effect on BCABC with cipher text =250 character


CR effect on BCABC with cipher text =500 character


CR effect on BCABC with cipher text =750 character

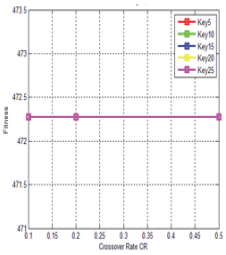
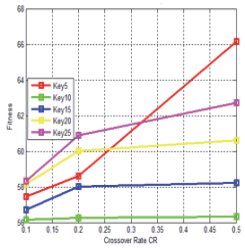
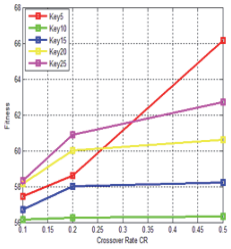
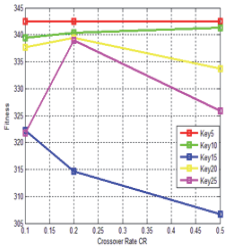

CR effect on BCABC with cipher text =1000 character

Figure 6 The best values obtained with different *CR* parameter using different lengths of cipher text using BCABC algorithm


CR effect on BCABC with cipher text =250 character


CR effect on BCABC with cipher text =500 character


CR effect on BCABC with cipher text =750 character

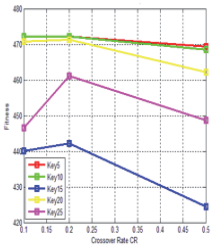

CR effect on BCABC with cipher text =1000 character

Figure 7 The mean values obtained with different *CR* parameter using different lengths of cipher text using BCABC algorithm

Table 3 The number of key letters recovered correctly from different length of English cipher text via ABC algorithm, NKR: The number of key letters recovered correctly.

Cipher text length	Key Size	Minimum of NKR	Maximum of NKR	Mean of NKR	Standard deviation of NKR	Minimum fitness	Maximum fitness	Mean of fitness	Standard deviation of fitness
250	5	0	4	2.5333	0.8996	52.5114	64.0267	56.4020	2.7069
	10	0	0	0.0000	0.0000	50.5857	53.3889	51.5362	0.7267
	15	0	1	0.1000	0.3051	49.8574	53.0608	50.9561	0.8027
	20	0	1	0.0333	0.1826	49.0367	50.7284	49.8444	0.4811
	25	0	3	0.3667	0.7649	48.5052	51.7278	49.8536	0.9141
500	5	3	5	3.9667	0.4138	143.4218	205.8708	167.5250	12.4580
	10	3	5	4.2667	0.5208	130.4588	151.6807	138.8680	5.3378
	15	2	5	3.5333	0.9371	118.0250	129.3112	123.3281	3.2605
	20	1	6	4.2667	1.2576	110.1442	118.0029	114.1279	1.9824
	25	1	9	4.0667	1.5960	108.9989	127.9195	113.3820	3.6811
750	5	3	5	4.0333	0.3198	250.3419	342.4439	283.6307	17.2920
	10	4	7	4.9667	0.6687	219.5141	259.3519	234.3092	8.5797
	15	3	6	4.4333	0.6261	191.8620	225.1796	202.9465	6.7211
	20	3	7	4.7000	0.8769	181.6796	202.4107	188.8577	4.2668
	25	3	7	4.7333	0.9803	177.3535	189.7726	181.5711	3.3355
1000	5	4	5	4.1333	0.3457	380.3119	472.2756	400.8530	28.6436
	10	4	6	5.1000	0.6618	300.5131	359.7419	321.9401	14.3074
	15	4	7	4.6667	0.7581	268.1845	306.3119	282.6832	8.0079
	20	3	8	5.0333	1.2452	252.4580	282.1519	265.3535	7.0368
	25	3	7	5.5667	1.1943	247.9509	272.2407	254.6403	5.3756

Table 4 The number of key letters recovered correctly from different length of Turkish cipher text via ABC algorithm, NKR: The number of key letters recovered correctly

Cipher text length	Key Size	Minimum of NKR	Maximum of NKR	Mean of NKR	Standard deviation of NKR	Minimum fitness	Maximum fitness	Mean of fitness	Standard deviation of fitness
250	5	3	5	3.9667	0.4138	66.6567	87.6893	75.4893	4.7468
	10	0	5	2.2000	1.4479	54.7390	62.8773	57.0546	2.0074
	15	0	5	1.5667	1.6333	51.2920	59.3545	53.6795	1.6610
	20	0	5	2.4667	1.4077	50.6885	55.7378	52.9640	1.1948
	25	0	5	1.6000	1.5222	49.7042	56.7146	51.7996	1.4357
500	5	4	5	4.4667	0.5074	174.7537	208.7812	194.2237	14.1103
	10	4	6	5.0667	0.5833	138.3754	159.5398	146.1941	5.4336
	15	4	7	5.1667	0.7915	121.9972	136.7473	127.1416	3.4820
	20	2	7	4.6000	1.1626	117.4816	129.8581	123.3861	2.9602
	25	2	7	4.4667	1.4320	114.3958	122.3029	118.7351	2.1761
750	5	4	5	4.4333	0.5040	282.2977	332.4877	305.5313	24.0569
	10	4	6	5.1000	0.6074	215.8774	250.6668	231.1823	8.3609
	15	4	7	5.5333	0.8193	195.0175	217.4108	204.5360	5.2829
	20	3	8	5.2333	1.2229	190.1940	217.3559	196.5783	5.7820
	25	3	8	4.6667	1.0933	184.1261	194.2468	187.5224	2.5900
1000	5	4	5	4.5333	0.5074	384.5208	459.2877	427.2798	34.8339
	10	4	7	5.2667	0.6397	297.7808	353.7408	320.6774	13.5416
	15	4	7	5.2333	0.8172	272.4056	296.9508	284.0453	6.6736
	20	3	8	5.5000	1.2247	258.9538	278.1714	267.7404	4.5597
	25	4	8	5.2667	0.9803	252.1442	277.3108	260.0149	5.0598

Table 5 The number of key letters recovered correctly from different length of English cipher text via BCABC algorithm, NKR: The number of key letters recovered correctly

Cipher text length	Key Size	Minimum of NKR	Maximum of NKR	Mean of NKR	Standard deviation of NKR	Minimum fitness	Maximum fitness	Mean of fitness	Standard deviation of fitness
250	5	0	5	1.6000	1.7340	54.1025	86.6142	58.6266	6.9267
	10	0	0	0.0000	0.0000	53.9846	57.6774	56.2596	0.7367
	15	0	0	0.0000	0.0000	56.8034	60.7787	58.0226	0.9434
	20	0	0	0.0000	0.0000	56.5081	62.9036	60.0210	1.4739
	25	0	1	0.2667	0.4498	58.1778	64.0077	60.9069	1.3047
500	5	5	5	5.0000	0.0000	205.8708	205.8708	205.8708	0.0000
	10	8	10	9.5333	0.6814	168.3519	205.8708	199.2021	10.1564
	15	6	12	9.6667	1.6259	135.6376	186.1697	168.8796	13.2430
	20	4	19	12.8333	3.7516	126.8753	205.9255	165.6999	21.1167
	25	19	24	21.7667	1.5687	181.6719	207.9439	198.6383	6.6743
750	5	5	5	5.0000	0.0000	342.4439	342.4439	342.4439	0.0000
	10	9	10	9.9333	0.2537	311.7009	342.4439	340.4158	7.7185
	15	10	15	13.1000	1.3734	273.3619	342.4439	314.6871	19.9249
	20	18	20	19.7667	0.5040	320.8149	342.4439	339.4815	6.3839
	25	23	25	24.5667	0.7739	316.0656	342.4439	338.9197	7.0481
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	11	15	13.5667	1.1943	397.4719	472.2756	442.1973	23.9817
	20	19	20	19.9333	0.2537	456.8319	472.2756	471.3437	3.5673
	25	22	25	24.2000	0.9965	425.2619	472.2756	461.1489	14.2026

Table 6 The number of key letters recovered correctly from different length of Turkish cipher text via BCABC algorithm, NKR: The number of key letters recovered correctly.

Cipher text length	Key Size	Minimum of NKR	Maximum of NKR	Mean of NKR	Standard deviation of NKR	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	0	7	2.5667	2.6611	58.8406	81.0350	64.5094	6.0581
	15	0	9	3.4667	1.9070	60.4729	75.4298	67.5728	3.4528
	20	1	13	7.9000	3.1878	61.0676	83.5661	70.3603	5.6505
	25	3	13	7.9333	2.4202	64.3895	79.3419	71.1128	4.2436
500	5	5	5	5.0000	0.0000	208.7812	208.7812	208.7812	0.0000
	10	9	10	9.9333	0.2537	192.6834	208.7812	207.7491	3.9313
	15	15	15	15.0000	0.0000	208.7812	208.7812	208.7812	0.0000
	20	16	20	19.6333	0.8503	197.3783	208.7812	207.1740	3.2498
	25	22	25	23.7333	0.6915	200.2268	209.5443	208.5595	1.9624
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	14	15	14.9667	0.1826	316.1677	332.4877	331.9437	2.9796
	20	19	20	19.9333	0.2537	321.7077	332.4877	331.9257	2.2260
	25	24	25	24.7333	0.4498	323.7808	332.4877	331.4456	2.3060
1000	5	5	5	5.0000	0.0000	459.2877	459.2877	459.2877	0.0000
	10	10	10	10.0000	0.0000	459.2877	459.2877	459.2877	0.0000
	15	14	15	14.9000	0.3051	437.1108	459.2877	457.3342	6.0000
	20	18	20	19.8333	0.4611	431.1108	459.2877	456.7755	6.8726
	25	24	25	24.8000	0.4068	444.9677	459.2877	457.7653	3.6525

Table 7 Correct key-based comparison of algorithms for English texts

		English						Turkish					
Cipher text length	Key Size	Best value		Mean value		Standard deviation		Best value		Mean value		Standard deviation	
		ABC	BCABC	ABC	BCABC	ABC	BCABC	ABC	BCABC	ABC	BCABC	ABC	BCABC
250	5	4	5	2.5333	1.6000	0.8996	1.7340	5	5	3.9667	5.0000	0.4138	0.0000
	10	0	0	0.0000	0.0000	0.0000	0.0000	5	7	2.2000	2.5667	1.4479	2.6611
	15	1	0	0.1000	0.0000	0.3051	0.0000	5	9	1.5667	3.4667	1.6333	1.9070
	20	1	0	0.0333	0.0000	0.1826	0.0000	5	13	2.4667	7.9000	1.4077	3.1878
	25	3	1	0.3667	0.2667	0.7649	0.4498	5	13	1.6000	7.9333	1.5222	2.4202
500	5	5	5	3.9667	5.0000	0.4138	0.0000	5	5	4.4667	5.0000	0.5074	0.0000
	10	5	10	4.2667	9.5333	0.5208	0.6814	6	10	5.0667	9.9333	0.5833	0.2537
	15	5	12	3.5333	9.6667	0.9371	1.6259	7	15	5.1667	15.0000	0.7915	0.0000
	20	6	19	4.2667	12.8333	1.2576	3.7516	7	20	4.6000	19.6333	1.1626	0.8503
	25	9	24	4.0667	21.7667	1.5960	1.5687	7	25	4.4667	23.7333	1.4320	0.6915
750	5	5	5	4.0333	5.0000	0.3198	0.0000	5	5	4.4333	5.0000	0.5040	0.0000
	10	7	10	4.9667	9.9333	0.6687	0.2537	6	10	5.1000	10.0000	0.6074	0.0000
	15	6	15	4.4333	13.1000	0.6261	1.3734	7	15	5.5333	14.9667	0.8193	0.1826
	20	7	20	4.7000	19.7667	0.8769	0.5040	8	20	5.2333	19.9333	1.2229	0.2537
	25	7	25	4.7333	24.5667	0.9803	0.7739	8	25	4.6667	24.7333	1.0933	0.4498
1000	5	5	5	4.1333	5.0000	0.3457	0.0000	5	5	4.5333	5.0000	0.5074	0.0000
	10	6	10	5.1000	10.0000	0.6618	0.0000	7	10	5.2667	10.0000	0.6397	0.0000
	15	7	15	4.6667	13.5667	0.7581	1.1943	7	15	5.2333	14.9000	0.8172	0.3051
	20	8	20	5.0333	19.9333	1.2452	0.2537	8	20	5.5000	19.8333	1.2247	0.4611
	25	7	25	5.5667	24.2000	1.1943	0.9965	8	25	5.2667	24.8000	0.9803	0.4068

Table 8 Fitness-based comparison of algorithms for English and Turkish texts

		English						Turkish					
Cipher text length	Key Size	Best value		Mean value		Standard deviation		Best value		Mean value		Standard deviation	
		ABC	BCABC	ABC	BCABC	ABC	BCABC	ABC	BCABC	ABC	BCABC	ABC	BCABC
250	5	64.0267	86.6142	56.4020	58.6266	2.7069	6.9267	87.6893	87.6893	75.4893	87.6893	4.7468	0.0000
	10	53.3889	57.6774	51.5362	56.2596	0.7267	0.7367	62.8773	81.0350	57.0546	64.5094	2.0074	6.0581
	15	53.0608	60.7787	50.9561	58.0226	0.8027	0.9434	59.3545	75.4298	53.6795	67.5728	1.6610	3.4528
	20	50.7284	62.9036	49.8444	60.0210	0.4811	1.4739	55.7378	83.5661	52.9640	70.3603	1.1948	5.6505
	25	51.7278	64.0077	49.8536	60.9069	0.9141	1.3047	56.7146	79.3419	51.7996	71.1128	1.4357	4.2436
500	5	205.8708	205.8708	167.5250	205.8708	12.4580	0.0000	208.7812	208.7812	194.2237	208.7812	14.1103	0.0000
	10	151.6807	205.8708	138.8680	199.2021	5.3378	10.1564	159.5398	208.7812	146.1941	207.7491	5.4336	3.9313
	15	129.3112	186.1697	123.3281	168.8796	3.2605	13.2430	136.7473	208.7812	127.1416	208.7812	3.4820	0.0000
	20	118.0029	205.9255	114.1279	165.6999	1.9824	21.1167	129.8581	208.7812	123.3861	207.1740	2.9602	3.2498
	25	127.9195	207.9439	113.3820	198.6383	3.6811	6.6743	122.3029	209.5443	118.7351	208.5595	2.1761	1.9624
750	5	342.4439	342.4439	283.6307	342.4439	17.2920	0.0000	332.4877	332.4877	305.5313	332.4877	24.0569	0.0000
	10	259.3519	342.4439	234.3092	340.4158	8.5797	7.7185	250.6668	332.4877	231.1823	332.4877	8.3609	0.0000
	15	225.1796	342.4439	202.9465	314.6871	6.7211	19.9249	217.4108	332.4877	204.5360	331.9437	5.2829	2.9796
	20	202.4107	342.4439	188.8577	339.4815	4.2668	6.3839	217.3559	332.4877	196.5783	331.9257	5.7820	2.2260
	25	189.7726	342.4439	181.5711	338.9197	3.3355	7.0481	194.2468	332.4877	187.5224	331.4456	2.5900	2.3060
1000	5	472.2756	472.2756	400.8530	472.2756	28.6436	0.0000	459.2877	459.2877	427.2798	459.2877	34.8339	0.0000
	10	359.7419	472.2756	321.9401	472.2756	14.3074	0.0000	353.7408	459.2877	320.6774	459.2877	13.5416	0.0000
	15	306.3119	472.2756	282.6832	442.1973	8.0079	23.9817	296.9508	459.2877	284.0453	457.3342	6.6736	6.0000
	20	282.1519	472.2756	265.3535	471.3437	7.0368	3.5673	278.1714	459.2877	267.7404	456.7755	4.5597	6.8726
	25	272.2407	472.2756	254.6403	461.1489	5.3756	14.2026	277.3108	459.2877	260.0149	457.7653	5.0598	3.6525

Table 9 Statistical comparison results for English and Turkish cases

Cipher text length	Key Size	English		Turkish	
		p Values	*h* Values	*p* Values	*h* Values
		BCABC & ABC	BCABC & ABC	BCABC & ABC	BCABC & ABC
250	5	0.42457	0	1.581e−11	1 - BCABC
	10	2.9654e−11	1 - BCABC	2.6695e−09	1 - BCABC
	15	3.0199e−11	1 - BCABC	3.0199e−11	1 - BCABC
	20	3.0199e−11	1 - BCABC	3.0199e−11	1 - BCABC
	25	3.0199e−11	1 - BCABC	3.0199e−11	1 - BCABC
500	5	1.6373e−11	1 - BCABC	5.2692e−06	1 - BCABC
	10	1.4411e−11	1 - BCABC	2.3657e−12	1 - BCABC
	15	3.0104e−11	1 - BCABC	1.2118e−12	1 - BCABC
	20	3.0199e−11	1 - BCABC	7.8787e−12	1 - BCABC
	25	2.9972e−11	1 - BCABC	1.4325e−11	1 - BCABC
750	5	1.2428e−11	1 - BCABC	2.1691e−06	1 - BCABC
	10	2.3638e−12	1 - BCABC	1.2118e−12	1 - BCABC
	15	2.971e−11	1 - BCABC	1.7203e−12	1 - BCABC
	20	6.4789e−12	1 - BCABC	2.3657e−12	1 - BCABC
	25	9.3936e−12	1 - BCABC	9.3352e−12	1 - BCABC
1000	5	1.8741e−10	1 - BCABC	2.711e−05	1 - BCABC
	10	1.2118e−12	1 - BCABC	1.2118e−12	1 - BCABC
	15	2.8628e−11	1 - BCABC	3.1602e−12	1 - BCABC
	20	2.3657e−12	1 - BCABC	4.111e−12	1 - BCABC
	25	2.1128e−11	1 - BCABC	6.4697e−12	1 - BCABC

From Tab. 7 and Tab. 8, ABC algorithm is efficient when Vigenere cipher uses a smaller length of key while when dealing with longer key lengths, it is not so efficient. On the other hand, BCABC is more efficient than ABC, specifically when the key size is up to 25, with cipher text size greater than 250 characters. Through some instances, the number of key letters recovered correctly, or the maximum fitness value was equal; nonetheless, we performed a statistical analysis to see whether these differences are significant. Non-parametric Wilcoxon rank sum test were applied to check whether the algorithms have equal median values since the runs failed in normality tests. BCABC algorithm was selected as control algorithm and the p-values and h-values are calculated for all instances as in Tab. 9. For English texts, BCABC algorithm was found better in 19 cases and worse in 1 case while for Turkish texts, BCABC algorithm was better in all cases.

6 CONCLUSION

Because the Vigenere cryptosystem key space is highly dimensional, neither brute force is practical in real time cryptosystems nor statistical approaches are useful when the key size is large. The objective of this paper was to analyse the overall applicability of ABC algorithm and propose a more efficient BCABC algorithm as a cryptanalysis method of traditional crypto systems focusing on Vigenere cipher.

The results are examined using statistical tools and it can be concluded that ABC is successful in recovering the whole key of Vigenere Cipher for keys of small size while BCABC has the ability to recover all of the letters for keys up to 25 letters with cipher text of more than 250 characters. This shows that the binomial crossover integrated to ABC algorithm improved the exploitation capability of the algorithm and BCABC is superior in terms of the accuracy in cryptanalysis of Vigenere cipher compared to ABC algorithm. With regard to the frequency based fitness function, it is not efficient in short messages but is better applicable to longer messages. However, it can be said that BCABC increases the time complexity compared to basic ABC algorithm.

Tailoring new fitness functions efficient for every different key size, text length, and language used is our future work.

Acknowledgement

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

7 REFERENCES

[1] Denning, D. E. R. (n.d.). *Cryptography and data security*. Reading, Mass.: Addison-Wesley.

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

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

[4] Hilton, R. (2012). *Automated cryptanalysis of monoalphabetic substitution ciphers using stochastic optimization algorithms* (Doctoral dissertation, Department of Computer Science and Engineering, University of Colorado).

[5] Stallings, W. (2006). *Cryptography and network security: principles and practice: William Stallings*. Upper Saddle River, NJ: Pearson/Prentice Hall.

[6] Barr, T. H. & Lindquester, T. E. (2002). *Invitation to cryptology*. Upper Saddle River, NJ: Prentice Hall.

[7] Karaboga, D. (2005). *An idea based on honey bee swarm for numerical optimization*, *200*, 1-10. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

[8] Akay, B. & Karaboga, D. (2015). A survey on the applications of artificial bee colony in signal, image, and video processing. *Signal, Image and Video Processing*, *9*(4), 967-990. https://doi.org/10.1007/s11760-015-0758-4

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

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

[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. *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. *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. *Journal of 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 Computation and 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] Boryczka, U. & Dworak, K. (2014). Cryptanalysis of Transposition Cipher Using Evolutionary Algorithms. *Computational Collective Intelligence. Technologies and Applications Lecture Notes in Computer Science*, 623-632. https://doi.org/10.1007/978-3-319-11289-3_63

[25] 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

[26] 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

[27] 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

[28] LI, H. O. Y. E. A. N. (2015). Heuristic cryptanalysis of classical and modern ciphers. *Heuristic cryptanalysis of classical and modern ciphers*. Place of publication not identified: LAP LAMBERT ACADEMIC PUBL. https://doi.org/10.1109/ICON.2005.1635595

[29] Ali, I. K. & Mahmod, A. G. (2015). Hybrid Bees Algorithm With Simulated Annealing for Cryptanalysis of Simple Substitution Cipher. *Journal of University of Babylon*, *23*(2), 565-574.

[30] Din, M., Pal, S. K., & Muttoo, S. K. (2019). A Review of Computational Swarm Intelligence Techniques for Solving Crypto Problems. *Advances in Intelligent Systems and Computing Soft Computing: Theories and Applications*, 193-203. https://doi.org/10.1007/978-981-13-0589-4_18

[31] Civicioglu, P. & Besdok, E. (2011). A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. *Artificial Intelligence Review*, *39*(4), 315-346. https://doi.org/10.1007/s10462-011-9276-0

[32] Zaharie, D. (2009). Influence of crossover on the behavior of Differential Evolution Algorithms. *Applied SoftComputing*, *9*(3), 1126-1138. https://doi.org/10.1016/j.asoc.2009.02.012

[33] Weber, M. & Neri, F. (2012). Contiguous Binomial Crossover in Differential Evolution. *Swarm and Evolutionary Computation Lecture Notes in Computer Science*, 145-153. https://doi.org/10.1007/978-3-642-29353-5_17

[34] Islam, S. M., Das, S., Ghosh, S., Roy, S., & Suganthan, P. N. (2012). An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *42*(2), 482-500. https://doi.org/10.1109/TSMCB.2011.2167966

[35] Lin, C., Qing, A., & Feng, Q. (2010). A comparative study of crossover in differential evolution. *Journal of Heuristics*, *17*(6), 675-703. https://doi.org/10.1007/s10732-010-9151-1

[36] Yang, J., Li, W.-T., Shi, X.-W., Xin, L., & Yu, J.-F. (2013). A Hybrid ABC-DE Algorithm and Its Application for Time-Modulated Arrays Pattern Synthesis. *IEEE Transactions on Antennas and Propagation*, *61*(11), 5485-5495. https://doi.org/10.1109/TAP.2013.2279093

[37] Brajevic, I. (2015). Crossover-based artificial bee colony algorithm for constrained optimization problems. *Neural Computing and Applications*, *26*(7), 1587-1601. https://doi.org/10.1007/s00521-015-1826-y

[38] Yan, X., Zhu, Y., Zou, W., & Wang, L. (2012). A new approach for data clustering using hybrid artificial bee colony algorithm. *Neurocomputing*, *97*, 241-250. https://doi.org/10.1016/j.neucom.2012.04.025

[39] Liang, Z., Hu, K., Zhu, Q., & Zhu, Z. (2017). An enhanced artificial bee colony algorithm with adaptive differential operators. *Applied Soft Computing*, *58*, 480-494. https://doi.org/10.1016/j.asoc.2017.05.005

[40] Jadon, S. S., Tiwari, R., Sharma, H., & Bansal, J. C. (2017). Hybrid Artificial Bee Colony algorithm with Differential Evolution. *Applied Soft Computing*, *58*, 11-24. https://doi.org/10.1016/j.asoc.2017.04.018

[41] 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

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

[43] Dalkılıç, M. E. & Dalkılıç, G. (2002). On the Cryptographic Patterns and Frequencies in Turkish Language. *International Conference on Advances in Information Systems*, *2457*, ADVIS 2002. Lecture Notes in Computer Science, 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

Appendix:

Table A.1 English unigram frequencies are as follows (in percent %)

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

Table A.2 English bigram frequencies are as follows (in percent %)

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

Table A.3 Turkish unigram frequencies are as follows (in percent %)

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

Table A.4 Turkish bigram frequencies are as follows (in percent %)

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