



You have downloaded a document from
RE-BUŚ
repository of the University of Silesia in Katowice

Title: Breaking Data Encryption Standard with a Reduced Number of Rounds Using Metaheuristics Differential Cryptanalysis

Author: Kamil Dworak, Urszula Boryczka

Citation style: Dworak Kamil, Boryczka Urszula. (2021). Breaking Data Encryption Standard with a Reduced Number of Rounds Using Metaheuristics Differential Cryptanalysis. „Entropy” (2021, iss. 12, art. no. 1697, s. 1-21), DOI:10.3390/e23121697



Uznanie autorstwa - Licencja ta pozwala na kopiowanie, zmienianie, rozprowadzanie, przedstawianie i wykonywanie utworu jedynie pod warunkiem oznaczenia autorstwa.

Article

Breaking Data Encryption Standard with a Reduced Number of Rounds Using Metaheuristics Differential Cryptanalysis

Kamil Dworak *  and Urszula Boryczka 

Faculty of Science and Technology, University of Silesia in Katowice, Będzińska 39, 41-200 Sosnowiec, Poland; urszula.boryczka@us.edu.pl

* Correspondence: kamil.dworak@us.edu.pl

Abstract: This article presents the author's own metaheuristic cryptanalytic attack based on the use of differential cryptanalysis (DC) methods and memetic algorithms (MA) that improve the local search process through simulated annealing (SA). The suggested attack will be verified on a set of ciphertexts generated with the well-known DES (data encryption standard) reduced to six rounds. The aim of the attack is to guess the last encryption subkey, for each of the two characteristics Ω . Knowing the last subkey, it is possible to recreate the complete encryption key and thus decrypt the cryptogram. The suggested approach makes it possible to automatically reject solutions (keys) that represent the worst fitness function, owing to which we are able to significantly reduce the attack search space. The memetic algorithm (MASA) created in such a way will be compared with other metaheuristic techniques suggested in literature, in particular, with the genetic algorithm (NGA) and the classical differential cryptanalysis attack, in terms of consumption of memory and time needed to guess the key. The article also investigated the entropy of MASA and NGA attacks.

Keywords: differential cryptanalysis; metaheuristics; symmetric block ciphers; memetic algorithms; DES; simulated annealing



Citation: Dworak, K.; Boryczka, U. Breaking Data Encryption Standard with a Reduced Number of Rounds Using Metaheuristics Differential Cryptanalysis. *Entropy* **2021**, *23*, 1697. <https://doi.org/10.3390/e23121697>

Academic Editor: Masahito Hayashi

Received: 15 November 2021

Accepted: 15 December 2021

Published: 18 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The growing popularity of computerisation, and at the same time the Internet itself, results in a growing demand for more and more advanced security methods. Restrictions such as individual user access control or basic authentication have become insufficient today. For several decades, engineers concentrating on the topic of information security have designed special cryptographic algorithms that meet the most important security aspects.

The main assumption of cryptography is not to hide the fact of the existence of information, but to keep its real image secret. The message is transformed in such a way that it is readable only to its author and the recipient it is dedicated to [1,2].

Contemporary symmetric block ciphers implement the process of transformation of the plain text using the Feistel cipher and the generalized substitution-permutation network [2]. In 1990, a completely new cryptanalytical method was made public, namely differential cryptanalysis [3]. In the case of the most modern and advanced encryption algorithms, the differential cryptanalysis itself turns out to be ineffective. In order to improve the attack performance, it was proposed to combine metaheuristic algorithms with the differential cryptanalysis algorithm.

In general, metaheuristic algorithms are used to obtain approximate solutions. In the case of cryptanalysis, it is necessary to guess the ideal decryption key—an approximate solution is unacceptable. Due to the avalanche effect present in every encryption algorithm today, changing any bit at the input causes a complete mixing of all bits at the output, which in fact results in the generation of a completely new ciphertext [1]. The developed algorithm enables automatic sifting of the keys with the worst value of the fitness function, owing to which the set of potential solutions will be significantly reduced.

Additional analytical properties of memetic algorithms improve the local search process in such a way as to achieve the best solution in the shortest possible time.

Metaheuristic algorithms are more and more often used in computer science, and thus in the domain of computer security. In the literature, we can find publications describing all kinds of metaheuristic attacks targeting both classical ciphers, contemporary symmetric block ciphers and stream ciphers. A literature review of publications is presented in Table 1.

Table 1. Literature review of researches on metaheuristics cryptanalysis.

Year	Authors	Algorithm	Cipher
2007	Song et al. [4]	GA	Four-Round DES
2007	Tadros et al. [5]	GA	Four-Rounded DES
2009	Garg [6]	GA and MA	Simplified Data Encryption Standard (SDES)
2010	Hu [7]	GA	Tiny Encryption Algorithm (TEA)
2011	Abd-Elmonim [8]	PSO	DES
2011	Vimalathithan and Valarmathi [9]	GA, PSO and Genetic Swarm Optimization (GSO)	Simplified Data Encryption Standard (SDES)
2012	Jadon et al. [10]	Binary PSO	DES
2012	Pandey and Mishra [11]	PSO	DES
2013	Ali [12]	Bees algorithm	Substitution Ciphers
2014	Boryczka and Dworak [13]	EA	Transposition Cipher
2014	Mekhaznia and Menai [14]	ACO and PSO	Feistel, Vigenere, and substitution ciphers
2015	Bhateja et al. [15]	Cuckoo Search	Vigenere cipher
2015	Jain et al. [16]	Cuckoo Search	Substitution Ciphers
2016	Amic et al. [17]	Binary Firefly Algorithm	DES
2016	Dworak et al. [18]	GA and MA	Simplified Data Encryption Standard (SDES)
2016	Dworak and Boryczka [19]	EA	Four-Rounded Fast Data Encipherment Algorithm (FEAL)
2017	Amic et al. [20]	Binary Cat Swarm Optimization (BCSO)	DES
2017	Jain et al. [21]	Cuckoo Search	Knapsack Cryptosystem
2017	Dworak and Boryczka [22]	GA	Six-Rounded DES
2018	Polak and Boryczka [23]	Tabu Search	RC4 and VMPC
2019	Amic et al. [20]	Dolphin Swarm Algorithm (DSA)	DES
2019	Kamal et al. [24]	Binary Cuckoo Search	Simplified Data Encryption Standard (SDES)
2019	Polak and Boryczka [25]	Tabu Search	RC4+
2020	Sabonchi et al. [26]	DE, GA and PSO	Vigenere cipher
2021	Grari et al. [27]	ACO	Merkle-Hellman cipher

In [4], the authors focused on evolutionary cryptanalysis using GA on DES4 ciphers by comparing the same bits between original and encrypted ciphertexts. Tadros in [5] presented another GA used to break FEAL8 and DES4 ciphers. Garg in [6] included a comparison between MA and GA during cryptanalysis of SDES encryption algorithm relying on n-gram statistics and frequency analysis method. Another approach was present by Hu in [7], quantum-inspired GA has been applied to break TEA. Abd-Elmonim described another attack, based on the PSO algorithm, responsible to break the full 16-rounded DES cipher in [8]. Vimalathithan and Valarmathi presented their researches about combining the effectiveness of GA and PSO as a new Generic Swarm Optimization algorithm to attack SDES cipher. In 2012, Jadon [10] and Pandey, with Mishra published interesting approaches related to Binary PSO and original PSO algorithms used in cryptanalysis attacks dedicated to DES cipher.

In the following years, Ali [12], Mekhaznia and Menai [14], Bhateja [15], Jain [16,21], and Sabonchi [26] focused on cryptanalysis of classical ciphers such as substitution, transposition, and Vigenere ciphers using many popular metaheuristics like Bees, EA, ACO, PSO and Cuckoo Search algorithms.

Amic in [17,20,28] presented Binary Firefly, Binary Cat Swarm Optimisation (BCSO), and Dolphin Swarm (DSA) algorithm—all directed against DES cipher. In [24] Kamal described the Binary Cuckoo Search algorithm used on ciphertext generated by SDES cipher.

Polak and Boryczka presented new cryptanalysis attacks dedicated to another subset of encryption algorithms—stream ciphers (RC4, VMPC, and RC4+), using Tabu Search in [23,25]. In 2021, Grari [27] published ACO algorithm dedicated Markle-Hellman cipher.

The next chapter is dedicated to a brief introduction to symmetric block ciphers and the DES cipher. The third chapter presents the basic assumptions of differential cryptanalysis, which were used and which constituted a basis for the design work on the MASA algorithm. Chapter four contains a detailed description of the developed metaheuristic attack carried out with the use of MA. The next chapter focuses on describing the runtime environment, including presenting all the parameters selected for each attack. This chapter also presents the results of the experiments, including the entropy studies for the MASA and NGA algorithms. The second to last chapter presents a detailed analysis of the effectiveness of the attacks presented, both in terms of the number of proven solutions and the time of decryption of the cryptogram. The article is concluded with a brief summary of the various stages of the research. This chapter also suggests further research directions. Appendix A is attached to this article, detailing the results for the Ω_2 characteristic.

2. Symmetric Block Ciphers

Symmetric ciphers are still one of the most popular encryption algorithms. In this type of ciphers, only one main key is used, which simultaneously takes on a role of an encryption and decryption key, which can be written as $K_E = K_D$. In the case of block ciphers, each message is divided into a finite number of blocks of the same length—for example, 64-bit blocks. Then they are transferred to the appropriate encryption function. Exactly one block of the ciphertext is generated from one block of plain text. If the message cannot be divided into even blocks, an additional block is created to store the last, incomplete, fragment of data. Then, for consistency, it is supplemented with default values or zeros.

These algorithms are perfect for encrypting larger volumes of data stored, that is, in all kinds of warehouses, wholesalers or databases. The most popular block cipher schemes include ciphers such as: DES and AES.

Data Encryption Standard

The DES cipher has been designed in such a way that the avalanche effect occurs from the very beginning of the algorithm [1]. Changing any input bit forces us to change at least half, and sometimes even all, of the output bits. The state of each bit at the output depends on each bit specified at the input [29].

The basic version of the cipher converts 64-bit plain text blocks into 64-bit ciphertext blocks, using a 64-bit encryption key K [2,30]. After running the algorithm, the primary key is reduced to 56 bits by removing every eighth parity bit. K is then subjected to breaking into six 48-bit subkeys, used in each of the cipher rounds, K_1, \dots, K_6 —A description of the primary key distribution process is presented in detail in [1,2,29–32]. Figure 1 shows a 6-round DES algorithm.

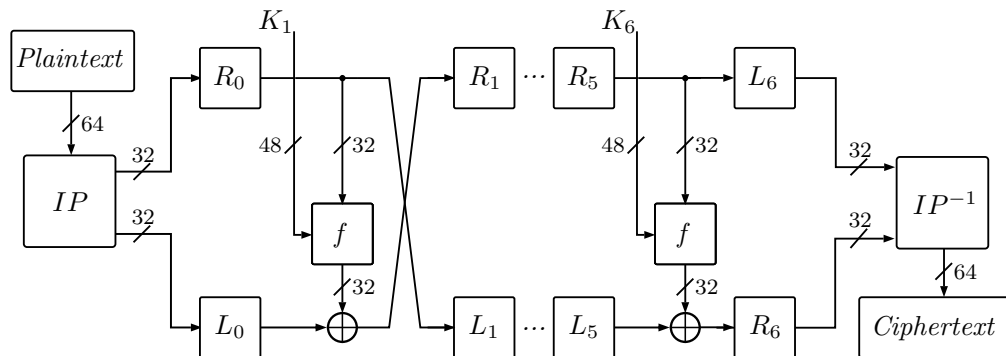


Figure 1. Simplified diagram of the six-rounded Data Encryption Standard DES algorithm.

The plain text block is passed to the initial IP permutation. Then, the generated block is divided into two regular 32-bit parts, R and L . In the next steps, six identical encryption cycles will be run, in which the right part of the R_i is passed to the f -round function along with the corresponding subkey K_i . Then, the generated data block is subjected to the exclusive disjunction operation with the left part of the L_i , resulting in a new right part of the R_{i+1} . The new left part of the L_{i+1} is copied from the right part of the previous R_i cycle.

After all the cipher rounds have been completed, parts of the L_6 and R_6 are combined into a 64-bit block, which will undergo the last transformation by the IP^{-1} inverse permutation function. The result of transposition of individual bits will be a 64-bit cryptogram block.

The f round function has been visualized in Figure 2. As an input parameter, a 32-bit data block is given, which at the very beginning will be extended via permutation E . The aim of this transformation is to align the length of the transferred block with the size of the subkey by duplicating the selected bits. By allowing one bit to influence two substitutions, the avalanche effect is increased [1]. The generated sequence is modulo two sum with subkey bits and then divided into eight 6-bit B_1 – B_8 blocks.

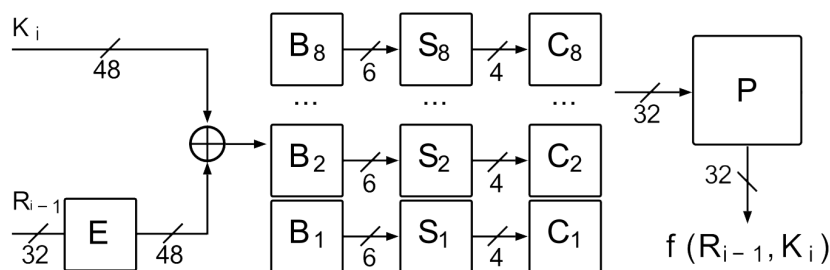


Figure 2. The f round function of the Data Encryption Standard DES algorithm. The only one, nonlinear, element of the DES cipher.

Each of the B_j blocks will be transferred to the so-called substitution matrix called S-blocks S_j . The main aim of this transform is to compress the input data. 6-bit data blocks will be converted into 4-bit blocks. S_j consist of integers between 0 and 15, stored in matrices of sixteen columns and four rows. The first and last bits of a 6-bit sequence B_j determine the line number. The remaining four bits represent the number of the column from which the return value will be selected [1,2,30].

S_j are the only nonlinear element of the DES standard. Changing one bit in an input sequence can lead to a complete mixing of all generated bits at the output. Modifications carried out in them have a significant impact on the level of complexity of cryptanalysis of the entire cipher. At the end of the f function, the generated sequences are combined into one 32-bit block, which will be passed to the permutation P —aimed at mapping each of the input bits to exactly one output bit without duplicating or omitting any of them [1].

3. Differential Cryptanalysis

The suggested algorithm is based on an attack with selected plain text. At the beginning, it should be assumed that the cryptanalyst has continuous access to the encryption algorithm, which allows him to select a pair of plain texts and analyse the generated ciphertexts. It is important that the tested pairs must differ from each other in a certain way. Most symmetric block ciphers determine this difference on the basis of a simple symmetric difference operation, which is written as $P' = P \oplus P^*$, where P and P^* are two crafted plain texts. Pairs may be generated in a pseudorandom way, although the most important condition is the difference P' , which must follow the established process. Next, the cryptanalyst checks how the determined difference changes in the subsequent phases of the cipher. Using the difference between the texts in individual iterations of the cipher, for a sufficiently large number of pairs, it is possible to assign different probabilities, suggesting the correctness of some subkeys [3]. When analyzing subsequent pairs of plain texts and ciphertexts, it turns out that one key may be more probable than the others.

Every modern cipher is non-linear—it means that it is not possible to find any pattern or rule by which to determine the value of a function for the next argument [3]. This non-linearity is obtained via the round f function. Each of all possible differences is characterized by a certain probability, which determines how often the f function returns the expected value [3]. These differences are called characteristics Ω . All possible characteristics can be determined by means of an additional matrix, where the rows correspond to all possible symmetric differences of the input blocks, and the columns to all possible symmetric differences of the output blocks [1]. Each of the elements will determine how many times the sum of the output bits occurs for the selected sum of the input bits.

By analysing the diagram shown in Figure 2, the input symmetric difference B' can be determined assuming that $E = E(R_{i-1})$:

$$B' = \parallel_{j=1}^8 B_j \oplus B_j^* = \parallel_{j=1}^8 (E_j(R_i) \oplus K_i) \oplus (E_j(R_i^*) \oplus K_i) = \parallel_{j=1}^8 E_j \oplus E_j^*, \quad (1)$$

where symbol \parallel stands for the concatenation of the successive data blocks. From the expression above, it can be seen that B' has nothing to do with the subkey. When the value of each B_j^* is known, the set of all ordered pairs (B_j, B_j^*) can be determined for the input symmetric difference as suggested in [31]:

$$\Delta(B_j') = \{(B_j, B_j \oplus B_j') : B_j \in (\mathbb{Z}_2)^6\}. \quad (2)$$

Knowing the output difference $C_j' = S_j(B_j) \oplus S_j(B_j^*)$, it becomes possible to generate the distribution of all possible input differences to all output differences according to the theorem described in [31]:

$$IN_j(B_j', C_j') = \{B_j \in (\mathbb{Z}_2)^6 : S_j(B_j) \oplus S_j(B_j \oplus B_j') = C_j'\}. \quad (3)$$

Most often, this distribution will be steady. The cryptanalyst's task is to find distributions that are as unsteady as possible. Based on the expression (3), an additional test set can be determined using the following formula [31]:

$$test_j(E_j, E_j^*, C_j') = \{B_j \oplus E_j : B_j \in IN_j(E_j', C_j')\}. \quad (4)$$

If the number of elements in $test_j$ is equal to the power of IN_j set, then the set must contain bits of the K_{ij} subkey [31].

This method makes it possible to restore the correct decryption key using 2^{47} selected plain texts and the corresponding ciphertexts.

4. Metaheuristics Differential Cryptanalysis

From the point of view of the developed attack, the IP and IP^{-1} permutations may be omitted. The algorithm begins by selecting the two most probable 3-round characteristics Ω_P^1 and Ω_P^2 mentioned in [31,32], which are presented in Figure 3, where P denotes characteristics for plaintext and C for ciphertexts.

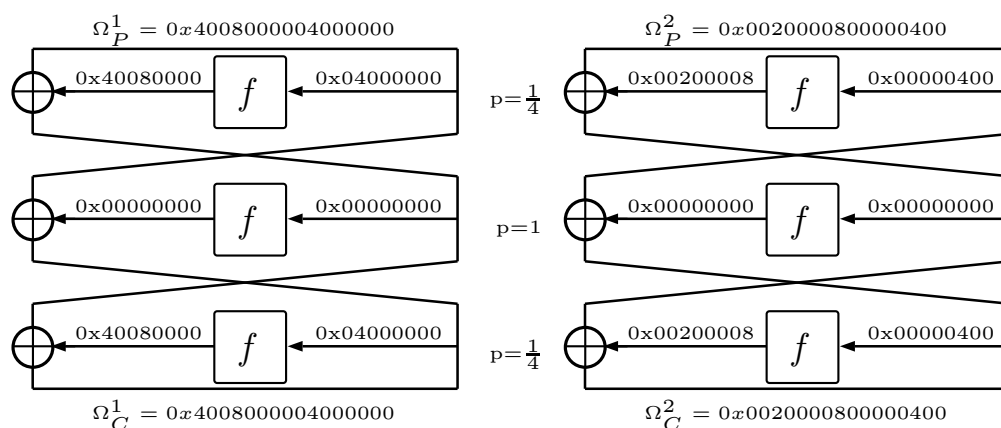


Figure 3. The two the most probable 3-round characteristics Ω_P^1 and Ω_P^2 for six rounded cipher DES [31,32].

The probability of each characteristic is exactly $P_\Omega = \frac{1}{16}$ in the fourth round of the encryption algorithm S-Blocks S_2, S_5, S_6, S_7, S_8 for Ω_P^1 and S_1, S_2, S_4, S_5, S_6 for Ω_P^2 for some input symmetric difference B'_j return an output symmetric difference C'_j equal to zero. Owing to this, it becomes possible to determine the sets $I_1 = \{2, 5, 6, 7, 8\}$ for Ω_P^1 and $I_2 = \{1, 2, 4, 5, 6\}$ for Ω_P^2 . The further description of the attack is identical for each of the characteristics Ω so it was decided to generalize it by introducing one generic I set consisting of elements of sets I_1 and I_2 .

The next step will be to generate a set of plain text pairs, along with a set of corresponding cryptograms, where the symmetrical difference will correspond to the characteristics Ω_1 and Ω_2 . The number of pairs needed is calculated using the signal-to-noise ratio [3]:

$$S/N = \frac{m \cdot p}{m \cdot \alpha \cdot \beta / 2^k} = \frac{2^k \cdot p}{\alpha \cdot \beta} = \frac{2^{30} \cdot 1 / 16}{4^5} = 2^{16}, \tag{5}$$

where:

- m —the number of pairs generated, having no effect on S/N ;
- p —the probability of the selected characteristic Ω ;
- k —number of bits of the subkey;
- α —the average number of subkeys, suggested by one pair;
- β —the ratio of the analysed pairs to all possible ones.

As suggested in [3], for $S/N = 2^{16}$, 7–8 correct pairs are needed for each of the characteristics. Due to the probability of P_Ω , a minimum of 150–200 pairs of plain text should be generated [3].

Additionally, the $test_j$ test set is determined, owing to which it will be possible to partially filter pairs from the set. If the power of the test set for at least one element from set I is equal to 0, the pair may be rejected:

$$\bigwedge_{j \in I} |test_j| > 0. \quad (6)$$

The aim of the suggested attack is to guess the last K_6 encryption subkey. If the difference of C' and part of R_5 is known, it becomes possible to analyze the various subkeys closely by comparing all bits of the output of the S-blocks with C' . A brute-force attack would need to check all 2^{30} solutions. MA can be used as an optimization tool that finds the correct solution in much shorter time.

Each individual is represented by a 30-bit K_j subkey. The fitness function is defined with the following formula:

$$F_f = \sum_{i=0}^n L - \sum_{j \in I} H((S_j(B_j) \oplus S_j(B_j^*)), P^{-1}(R'_6 \oplus L'_3)), \quad (7)$$

where:

- H —is the Hamming distance;
- L —the length of the subkey.

Owing to the knowledge of the probability of P_Ω , it is possible to estimate the value of L'_3 , while R'_6 can be obtained by analyzing a pair of generated ciphertexts. F_f counts the number of overlapping bits between the difference obtained from the S-blocks and the C' difference.

The algorithm uses standard one-point crossover. The locus is selected pseudorandomly from 1 to 30. The newly created subkeys can be modified with the use of a mutation operator—which consists in replacing two pseudorandomly selected bits. The algorithm selects individuals using tournament selection. A leader is elected from the set of all subkeys and it is passed to the crossover operator.

There is an additional local search process in the algorithm—it is performed using the simulated annealing algorithm. The MASA attack pseudocode for the ΩP characteristic is shown below. Due to the complexity of this algorithm, it was decided to divide it into two parts:

- the first one, Algorithm 1—responsible for generating a set of filtered pairs of plain text, ciphertexts and determining the $test_j$ test set for each of the indexes;
- the second one, presented in Algorithm 2—describing the memetic algorithm, along with the processes of selection, crossing, mutation and exploitation, taking into account the pseudocode of the basic simulated annealing algorithm.

Algorithm 1: The pseudocode of the set of pairs preparation process for the MASA attack.

```

1  $\Omega_p := \text{find\_most\_probabilistic\_characteristic}()$ 
2  $I := \text{determine\_set\_of\_indexes}()$ 
3  $set\_of\_pairs := \text{generate\_set\_of\_plaintext\_and\_ciphertext\_pairs}()$ 
4 for  $i := 0$  to  $size(set\_of\_pairs)$  do
5    $pair := set\_of\_pairs[i]$ 
6   foreach  $j \in I$  do
7      $test_j := \text{determine\_test\_set}(pair)$ 
8     if  $|test_j| == 0$  then
9        $\text{filter\_invalid\_pair}(set\_of\_pairs, pair)$ 
10      break
11    end
12  end
13 end

```

Running the MASA algorithm for Ω_p^1 will make it possible to guess 30 out of 48 bits of the K_6 subkey. Re-running the algorithm, this time for Ω_p^2 , allows us to find an extra 12 bits. In order to obtain the remaining 6 bits of the last K_6 subkey—coming from the S-block S_3 , we can use the brute-force method. Having the K_6 subkey, it is possible to recover 48 out of 56 bits of the decryption key by reversing the key decomposition process. The remaining 8 bits can be guessed using the brute-force method once again—for example, a brute force attack.

Algorithm 2: MASA attack pseudocode.

```

1  $P(0) := \text{create\_initial\_population}()$ 
2 for  $i := 0$  to  $\text{number\_of\_iterations}$  do
3    $\text{calculate\_fitness\_function\_value\_for\_each\_individual}()$ 
4   for  $j := 0$  to  $\text{population\_size}$  do
5      $\text{parent}_A := \text{tournament\_selection}()$ 
6      $\text{parent}_B := \text{tournament\_selection}()$ 
7      $\text{offspring} := [\text{parent}_A, \text{parent}_B]$ 
8     if  $\text{random}(0, 1) \geq \text{crossover\_probability}$  then
9        $\text{child}_A, \text{child}_B := \text{crossover}(\text{parent}_A, \text{parent}_B)$ 
10      if  $\text{random}(0, 1) \geq \text{mutation\_probability}$  then
11         $\text{child}_A := \text{mutation}(\text{child}_A)$ 
12      end
13
14      if  $\text{random}(0, 1) \geq \text{mutation\_probability}$  then
15         $\text{child}_B := \text{mutation}(\text{child}_B)$ 
16      end
17       $\text{offspring} := [\text{child}_A, \text{child}_B]$ 
18    end
19
20    foreach  $\text{child} \in \text{offspring}$  do
21       $T = T_0$ 
22      while  $T \geq T_{\text{MIN}}$  do
23         $\text{new\_child} := \text{change\_random\_bit}(\text{child})$ 
24         $\text{difference} := \text{new\_child.fitness} - \text{child.fitness}$ 
25        if  $\text{difference} > 0$  or
26           $\text{probability\_fun}(\text{difference}, T) > \text{random}(0, 1)$  then
27             $\text{child} := \text{new\_child}$ 
28          end
29           $T = T \cdot \alpha$ 
30        end
31      end
32    end
33 end

```

5. Experimental Results

This chapter describes the analysis of the proposed memetic attack MASA and NGA in terms of the quality and number of solutions obtained [22]. It was important to check whether the suggested algorithms make it possible to improve the time of finding the correct subkey. Another important aspect was to check whether the MASA memetic algorithm enables a more effective, and therefore more successful, differential cryptanalysis.

5.1. Selecting Parameters

As part of the experiments, the impact of the parameters listed below for each of the attacks on the convergence of the algorithm and the quality of the obtained solutions was examined:

- number of iterations for the MASA and NGA algorithms;
- population size for the MASA i NGA algorithms;
- number of plaintext and ciphertext pairs γ for the MASA and NGA algorithms;
- probability of the heuristic negation P_n for the NGA algorithm.

In the conducted experiments, the parameter values were used in various combinations and for the subsequent experiments, potentially the best values in terms of the running time of the algorithm were established. For the MASA memetic algorithm, the parameters were set according to Table 2 below:

Table 2. Parameters of the MASA algorithm.

Id	Parameter	Symbol	Value
1	Maximum number of iterations	It_{MAX}	100
2	Population size	N	10
3	Number of plaintext pairs	γ	200
4	Tourney size	T_{SIZE}	10
5	Crossover probability	P_c	0.9
6	Mutation probability	P_m	0.02
7	Initial temperature	T_0	1
8	Minimal temperature	T_{MIN}	0.1
9	Cooling rate	α	0.9

The description of the NGA algorithm parameters has been described in detail in the publication [19]. Table 3 presents the most important parameters of the NGA algorithm:

Table 3. Parameters of the NGA algorithm.

Id	Nazwa	Symbol	Value
1	Maximum number of iterations	It_{MAX}	100
2	Population size	N	10
3	Number of plaintext pairs	γ	200
4	Tourney size	T_{SIZE}	10
5	Crossover probability	P_c	0.9
6	Mutation probability	P_m	0.02
7	Heuristic operator probability	P_n	0.25

As was mentioned before, for the purposes of the tests, a simplified version of the DES cipher was used, in which the number of rounds was limited from 16 to 6. All other processes in the encryption algorithm, such as subkey generation and S-block compression, remained unchanged.

5.2. Comparative Study

Each of the algorithms was tested 30 times for each of the characteristics Ω . Table 4 below shows the value of the F_f fitness function for the MASA and NGA algorithms for the first characteristic Ω_1 . The remaining results—for the characteristic Ω_2 are given in Appendix A in the Table A1.

Table 4. Fitness function values for MASA and NGA algorithms for characteristic Ω_P^1 .

ID	MASA					NGA				
	Min	Med	Avg	Max	Std. Dev.	Min	Med	Avg	Max	Std. Dev.
1	885	953	95.5	1014	39.9	982	993	994.7	1014	11.5
2	929	997	989.2	1014	30.8	899	947	960.3	1012	40.3
3	888	935	948.7	1012	49.2	916	992	977.4	1014	37.8
4	978	1010	1003.9	1014	12.3	886	945	943.0	997	33.5
5	910	950	960.9	1014	38.9	922	978	982.3	1014	25.5
6	915	971	971.6	1014	35.1	871	978	960.1	1014	53.2
7	877	925	953.7	1014	52.6	928	998	990.1	1014	30.3
8	920	982	983.6	1014	31.1	900	960	958.9	1012	36.2
9	895	997	978.6	1014	35.2	943	980	981.3	1012	20.7
10	949	957	981.0	1014	30.1	863	934	945.9	1014	50.5
11	938	1014	996.8	1014	25.5	921	974	973.0	997	27.6
12	947	995	988.6	1014	22.1	899	975	965.1	1014	42.1
13	903	936	952.0	1014	36.9	891	978	962.3	1014	48.7
14	886	997	975.5	1014	46.6	855	991	958.0	1014	55.8
15	960	990	992.3	1014	20.0	881	920	951.3	1014	52.3
16	892	996	970.6	1014	42.3	884	998	978.0	1014	45.4
17	880	984	960.8	1014	50.1	911	954	962.5	1012	29.2
18	983	1014	1008.8	1014	10.0	865	978	958.1	1014	54.5
19	893	992	975.8	1014	38.1	878	951	951.3	998	44.0
20	956	1010	1003.3	1014	17.7	922	990	977.3	1014	34.1
21	892	979	965.9	998	38.3	875	929	945.5	1010	45.3
22	962	1009	1003.6	1014	15.1	909	1014	990.2	1014	38.2
23	885	939	960.7	1014	41.2	940	981	978.3	1010	23.2
24	901	970	962.2	1014	40.6	872	935	936.3	988	41.9
25	864	949	949.9	1014	50.3	890	954	944.9	988	34.9
26	958	992	991.8	1014	15.9	931	955	972.2	1014	28.4
27	899	920	949.2	1014	44.8	888	965	964.5	1014	43.2
28	902	966	965.4	1014	40.2	893	957	959.4	1014	40.6
29	971	997	999.2	1014	13.0	912	980	973.1	1014	37.2
30	922	997	977.2	1014	36.0	953	976	987.7	1014	21.2

Experiments in which the correct decryption key could not be guessed were marked in bold in the table above.

The probability of each of the characteristics for this cipher is not 100%. It means that despite striving for the maximum value of the fitness function, it will never be achieved. The inability to obtain the maximum value means that we are not able to terminate the running of the algorithm earlier than after the completion of all predetermined iterations.

Figures 4 and 5 present a list of all correctly guessed bits of the K_6 subkey for the MASA and NGA algorithms for the Ω_1 characteristic. The remaining results—for the Ω_2 characteristic are present in Appendix A in the Figures A1 and A2.

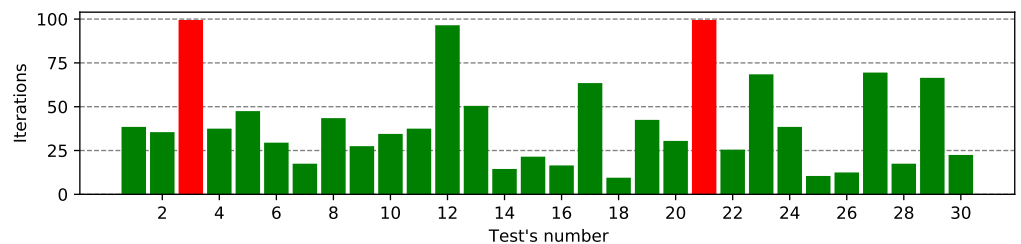


Figure 4. List of correctly guessed bits of MASA attack for the Ω_1 characteristic.

In a large number of cases, the MASA attack finds the correct subkey in the first 25 iterations. In approximately 6–7 cases, the algorithm found a solution using half of the

available iterations, while in the other two cases (tests #3 and #21, marked as red on the figure) the attack failed to cope with the given ciphertext. The algorithm found the correct decryption subkey in 93% of the cases - markes as green on the figure.

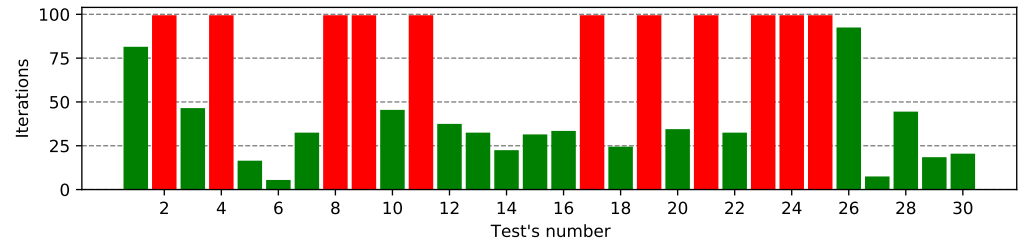


Figure 5. List of correctly guessed bits of NGA attack for the Ω_1 characteristic.

In the case of the NGA algorithm, the cipher was not cracked 11 times—which is over 37% of all possible approaches—red bars on the figure. During the remaining 63% of the tests, it was possible to crack the cipher with the decryption algorithm—green color. In most cases, it was possible to guess the correct subkey using only 30–40 iterations. The tests with identifiers #1 and #26 also deserve special attention. They show a very large number of iterations (over 80), which means that the NGA algorithm found the correct solution at the very end of its running.

On the presented bar plots we can notice the MASA algorithm is much effective because it successfully found the correct subkey in almost every test when NGA attack has worked in only 63% of experiments. Simulated annealing, used as an additional exploitation step of the MA, is more effective than the heuristic negation operator used in the NGA attack.

The next stage of the experiments was to analyze the course of the fitness function value using the convergence diagrams, which were presented successively, for the MASA attack and Ω_1 in Figures 6 and 7, for the NGA algorithm. Convergence diagrams for the Ω_2 were present in Appendix A in the Figure A3, for the MASA algorithm, and Figure A4 for the NGA attack.

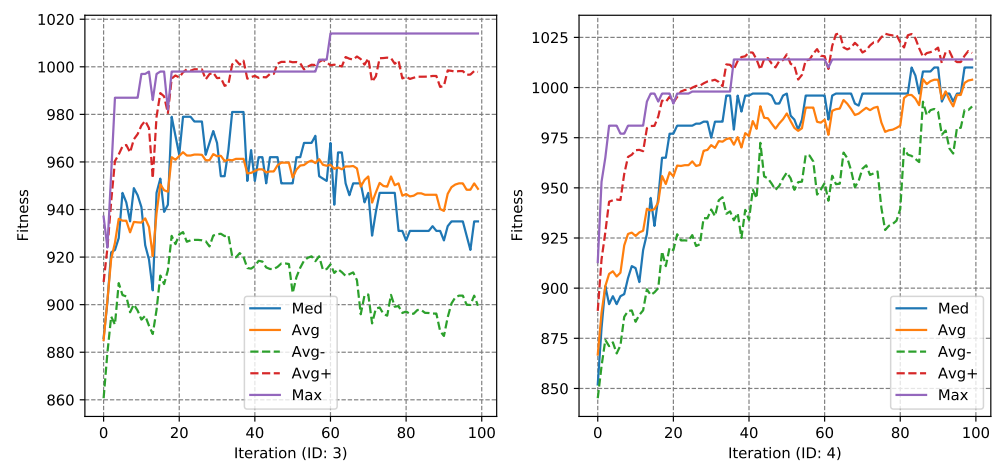


Figure 6. The MASA fitness function F_f convergence diagrams for Ω_1 (tests #3 and #4).

The above graph shows tests #3 and #4 with minimum, maximum, medians and averages—and average values increased and decreased by the standard deviation of the fitness function. The tests were selected in such a way as to visualize both a positive case—when it was possible to guess the correct subkey, and a negative one.

In the case of both tests of the MASA algorithm, a rapid increase in the maximum value of F_f can be noticed at the very beginning of the algorithm’s running. In further iterations, there are single drops of this value, after which the maximum value is stabilized

and then increased again. The median for 60% of the algorithm’s running time remains similar, only at the very end of its running we can notice its decrease. When analyzing the case #4 diagram, already in the first iterations of the algorithm, a rapid increase in the median value can be observed—the majority of individuals in the population have a similar value of the fitness function. This may be related to the algorithm falling into the local extreme, which it has not managed to leave.

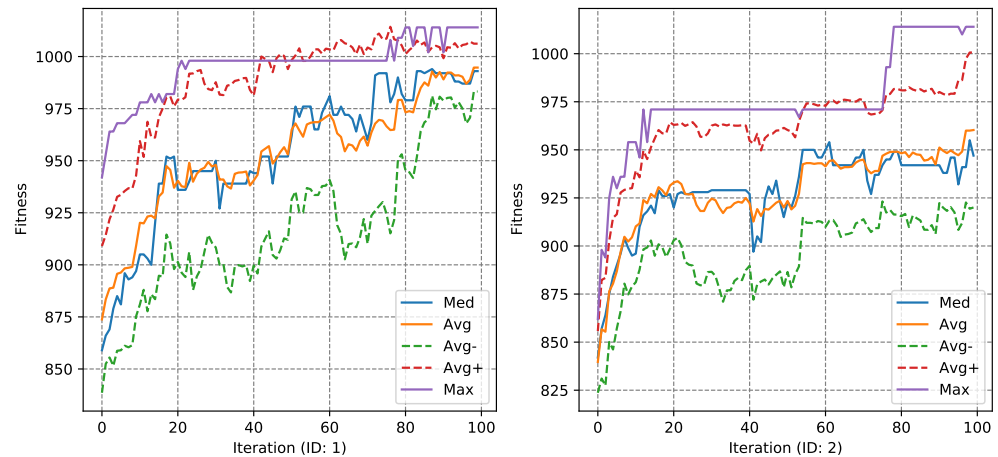


Figure 7. The NGA fitness function F_f convergence diagrams for Ω_1 (tests #1 and #2).

The next stage of the tests was to review the distribution of the fitness function values in the last iteration of each attack—the distribution is presented in Figure 8 for the MASA algorithm, and Figure 9 in the case of an NGA attack. Boxplots for the Ω_2 characteristic were present in Appendix A in the Figure A5, for the MASA algorithm, and Figure A6 for the NGA attack.

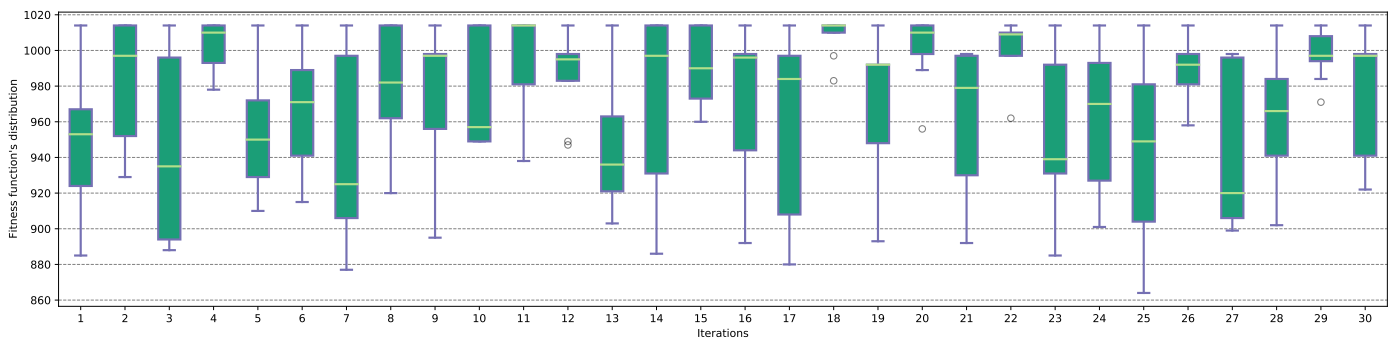


Figure 8. The distribution of the fitness function F_f values in the last iteration for the MASA algorithm and Ω_1 characteristic.

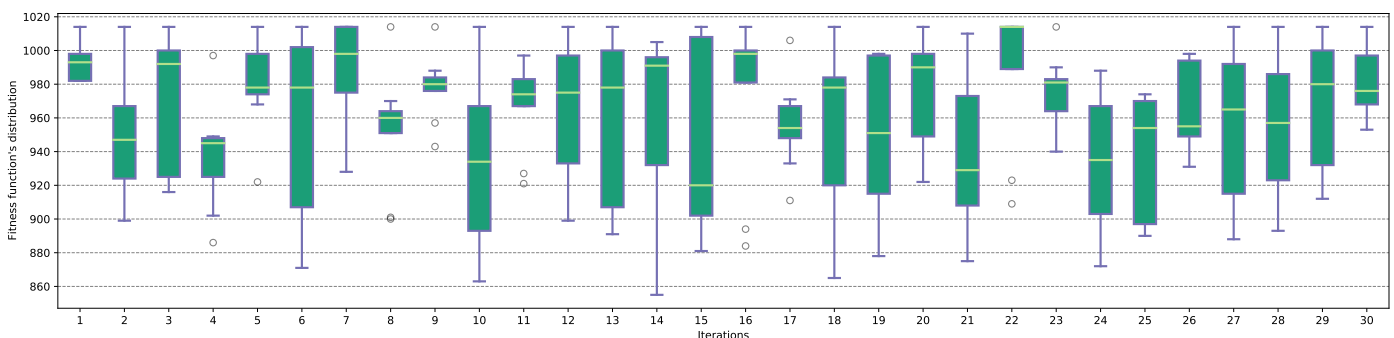


Figure 9. The distribution of the fitness function F_f values in the last iteration for the NGA algorithm and Ω_1 characteristic.

In the case of the MASA algorithm, some of the tests—for example, #18, #20 or #22—are characterized by a high degree of homogeneity, which means that the population is characterized by a low diversity of individuals. When analyzing each of the attacks, a large degree of variability between individuals can be observed, which is undoubtedly indicated by the median value, changing its position between the first and the third quartiles. In the case of the NGA algorithm, in some experiments, an unexpected increase of the value of the fitness function can be observed at the very end of the algorithm’s running—it is evidenced by the presence of the outlier of the maximum value.

The MASA and NGA attacks are characterized by a certain degree of pseudo-randomness. In order to perform statistical verification of the algorithms, a non-parametric Wilcoxon’s test was used to compare the results. The hypothesis H_0 , specifying no difference when comparing the samples, and the hypothesis H_1 , assuming a difference between the two samples, were set. The following criteria were used to perform the test:

- value of the fitness function—performed for the best quality subkeys found for each run;
- number of subkeys checked.

The weight of each criterion was expressed at the same value, set to 0.5. For the analyses performed, hypothesis H_0 was rejected at $p < 0.05$ —thus indicating the statistically important differences between the best results retrieved. The results obtained through the MASA algorithm are significantly better than the NGA attack.

5.3. Entropy Study

The possibility to maintain a highly diverse population may improve the algorithm’s ability not to fall into local extremes. In order to estimate the size of the disorder in the system, the entropy was used:

$$H(X) = \sum_{i=1}^n p(x_i) \log_2 \frac{1}{p(x_i)} = - \sum_{i=1}^n p(x_i) \log_2 p(x_i). \tag{8}$$

The entropy was computed by comparing the respective bits of each subkey with the corresponding bits of the best-adapted individual. An example for the population $P = \{11101, 10101, 11011, 11110\}$, where the last individual 11110 is the leader, is presented below (Table 5):

Table 5. Example scenario of the entropy calculation.

Subkey	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5
A	1	1	1	0	1
B	1	0	1	0	1
C	1	1	0	1	1
Leader	1	1	1	1	0
$p(x_1)$	1	0.75	0.75	0.50	0.25
$p(x_2)$	0	0.25	0.25	0.50	0.75
$H(x_1)$	$4 \cdot \log_2(1)$	$3 \cdot \frac{3}{4} \log_2(\frac{3}{4})$	$3 \cdot \frac{3}{4} \log_2(\frac{3}{4})$	$2 \cdot \frac{1}{2} \log_2(\frac{1}{2})$	$\frac{1}{4} \log_2(\frac{1}{4})$
$H(x_2)$	0	$\frac{1}{4} \log_2(\frac{1}{4})$	$\frac{1}{4} \log_2(\frac{1}{4})$	$2 \cdot \frac{1}{2} \log_2(\frac{1}{2})$	$3 \cdot \frac{3}{4} \log_2(\frac{3}{4})$

where:

- $p(x_1)$ —the probability of an identical bit occurring in a given position between individuals and the leader;
- $p(x_2)$ —the probability of a different bit occurring in a given position between individuals and the leader;
- $H(x_1)$ —entropy values for the probability $p(x_1)$, at a given position;
- $H(x_2)$ —entropy values for the probability $p(x_2)$, at a given position.

Based on the example listed in Table 5, the entropy value of the entire system can be computed as follows:

$$H = -(0 + 0 - 0.93 - 0.5 - 0.93 - 0.5 - 1 - 1 - 0.5 - 0.93) = 6.29. \quad (9)$$

Entropy for the MASA and NGA algorithms was visualized respectively in Figures 10 and 11. The charts show the maximum, minimum and average values. Moreover, it was decided to visualize the average value of entropy for both attacks on one graph, which is presented in Figure 12. The remaining results—for the characteristic Ω_2 are given in Appendix A in Figures A7–A9.

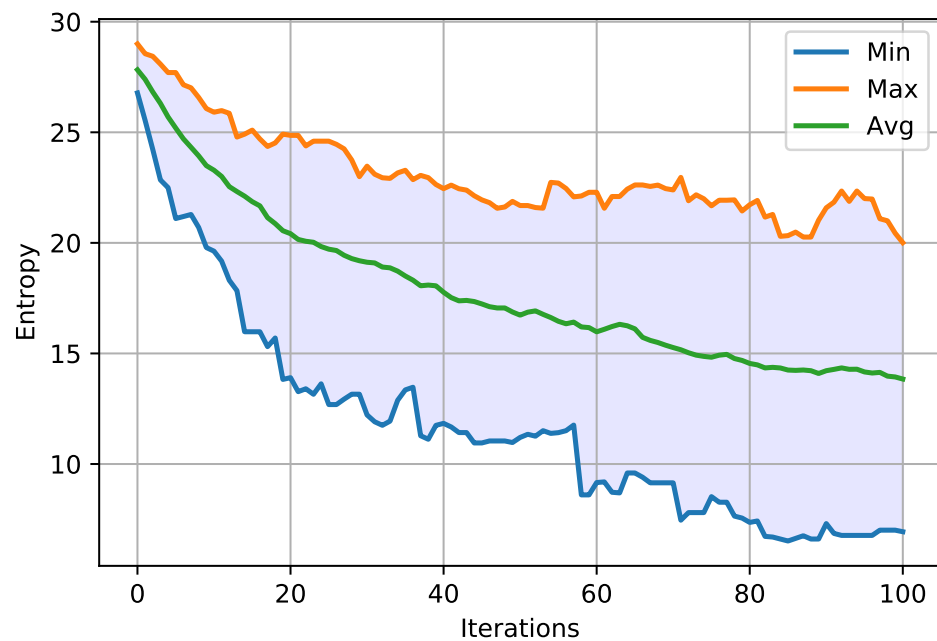


Figure 10. Minimum, maximum and average entropy, during all iterations, for MASA algorithm and Ω_1 characteristic.

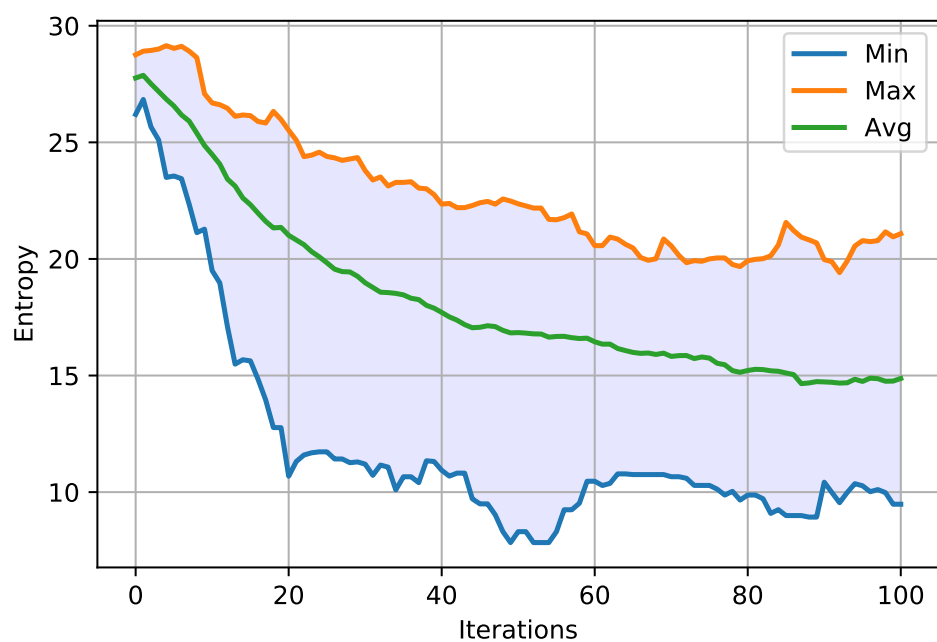


Figure 11. Minimum, maximum and average entropy, during all iterations, for NGA algorithm and Ω_1 characteristic.

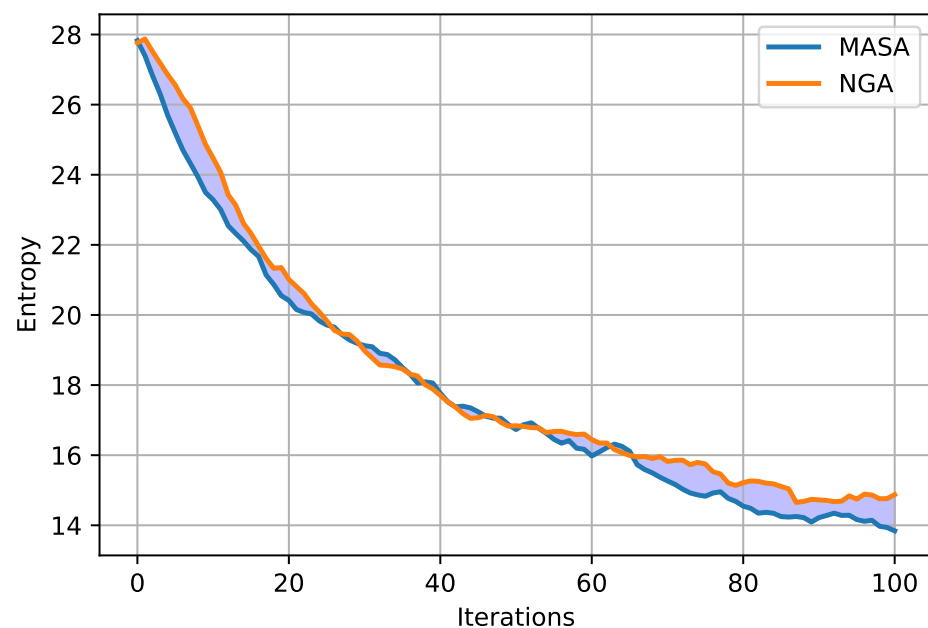


Figure 12. The comparison of the entropy of the MASA and NGA algorithms for the Ω_1 characteristic.

The entropy value was computed during each iteration and 30 launches of MASA and NGA attacks. During all the conducted tests, identical pairs of plain text and the corresponding ciphertexts were used, as well as the same encryption key—owing to which it was possible to make the most reliable comparison.

When analyzing the graphs presented above (Figures 10 and 11), a decrease in the entropy value can be noticed from the very beginning of the running of each of the algorithms. In the last iterations, a gradual stabilization of the system becomes visible, which would most probably be more noticeable after increasing the number of iterations. Comparing the average courses, it can be noticed in Figure 12 that the entropy value for the MASA attack is lower from the very beginning. Only from about the thirtieth iteration, the NGA algorithm obtains a similar value, and sometimes even lower, in relation to the MASA attack. Eventually, the entropy values for the NGA algorithm begin to stabilize at around the sixtieth iteration, while in the case of the MASA attack it continues to decrease. At the end of the algorithms' running, the difference in entropy value between attacks becomes visible.

The experiments carried out and described above clearly confirm the effectiveness of the suggested MASA attack, based on the use of memetic algorithms and simulated annealing. This information may be important during the running of the algorithm, since the probability of leaving the local extremum will be higher, and thus the quality of the final results will be better.

6. Conclusions

The article presents the results for the NGA genetic algorithm enriched with an additional heuristic negation operator and the MASA memetic algorithm that performs the local search process through simulated annealing. Both algorithms undoubtedly improve the process of an attack of differential cryptanalysis against the ciphertexts generated with the DES standard. An important aspect is the attempt to minimize the number of verified subkeys, which is presented in the table below:

The developed algorithms improve the effectiveness and efficiency of the attack, which is extremely important from the point of view of a cryptanalyst. Presented metaheuristics cryptanalysis, based on the differential cryptanalysis approach, can be helpful to raise the security level in already implemented IT systems. It can also be used to improve the complexity of ciphers at the design level. Proposed attacks, verified on the DES cipher, can be tested on more complicated modern encryption algorithms like AES or GOST ciphers.

Based on the tests presented in the previous section and Table 6, it is possible to clearly state the superiority of the MASA attack and the NGA algorithm over the classic differential cryptanalysis attack, due to the frequency of correctly guessed subkey and the number of proven solutions.

Table 6. Comparison of checked subkeys between MASA, NGA and differential cryptanalysis attacks.

Attack	Total Number of Checked Subkeys	Average Number of Checked Subkeys
MASA algorithm		
Ω_1	687,752	22,925.1
Ω_2	687,788	22,926.3
Σ	1,375,540	45,851.3
NGA algorithm		
Ω_1	252,456	8415.2
Ω_2	252,899	8430.0
Σ	505,355	16,845.2
Differential Cryptanalysis		
Ω_1	$30 \cdot (6 \cdot 2^{30} + 1024)$	$6 \cdot 2^{30} + 1024$
Ω_2	$30 \cdot (6 \cdot 2^{30} + 1024)$	$6 \cdot 2^{30} + 1024$
Σ	$30 \cdot (12 \cdot 2^{30} + 1024)$	$12 \cdot 2^{30} + 1024$

There are many parameters that influence the quality of offered solutions. Analyzing the importance of individual parameters, we intend in the future to conduct an analysis based on removing some of them or replacing them with a simplified version, without losing the quality of the offered solutions. Such approach (an ablation study) is very common when estimating costs of deep learning solutions and we hope that it will also be very effective here.

Work is currently underway on modifications of the developed attack, which would enable an even faster exploration of the solution space. In the future, an adaptive version of the memetic algorithm is expected to be developed to automatically adjust the attack parameters. A parallel implementation is also planned, which should be much more effective.

Simplified and the original DES encryption algorithms are commonly used by many cryptanalysts as a starting point to perform research and experimental studies in this discipline of science. It can be found in the literature review, presented in Table 1, in the introduction section. The authors of this article decided to use a reduced DES cipher for the purposes of developing new metaheuristic attacks described in the paper. Starting experiments from modern ciphers could be too complicated and significantly extend the research process. At the current state, we can test the proposed algorithms against more advanced symmetric block ciphers such as Twofish, AES, or GOST, which will definitely be the next step in future works.

Author Contributions: Conceptualization, K.D. and U.B.; formal analysis, K.D. and U.B.; investigation, K.D. and U.B.; methodology, K.D. and U.B.; project administration, K.D.; resources, K.D.; software, K.D.; supervision, K.D.; validation, K.D. and U.B.; visualization, K.D.; writing, original draft, K.D. and U.B.; writing, review and editing, K.D. and U.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this paper:

ACO	Ant Colony Optimization
BCSO	Binary Cat Swarm Optimization
DC	Differential Cryptanalysis
DES	Data Encryption Standard
DSA	Dolphin Swarm Algorithm
EA	Evolutionary Algorithms
FEAL	Fast Data Encipherment Algorithm
GSO	Genetic Swarm Optimization
MA	Memetic Algorithms
MASA	Memetic Algorithm Simulated Annealing
NGA	Negation Genetic Algorithms
PSO	Particle Swarm Optimization
RC4	Rivest Cipher 4
SDES	Simplified Data Encryption Standard
TEA	Tiny Encryption Algorithm
VMPC	Variably Modified Permutation Composition

Appendix A. The Comparative and Entropy Studies for the Ω_2 Characteristic

Table A1. Fitness function values for MASA and NGA algorithms for characteristic Ω_p^2 .

ID	MASA					NGA				
	Min	Med	Avg	Max	Std. Dev.	Min	Med	Avg	Max	Std. Dev.
1	906	984	1002.6	1095	57.1	987	1027	1041.5	1095	38.0
2	967	1059	1043.0	1095	42.6	1008	1034	1043.3	1074	21.3
3	946	973	997.8	1095	50.7	993	1010	1025.2	1075	29.2
4	1008	1041	1048.5	1095	29.2	968	1047	1046.8	1074	31.5
5	1008	1044	1045.6	1095	20.2	922	1021	1024.3	1095	52.7
6	943	989	1018.7	1095	60.7	964	1044	1043.5	1074	35.4
7	959	1041	1038.8	1095	40.0	888	957	994.0	1095	72.2
8	940	1011	1020.6	1095	44.0	953	1059	1036.3	1095	49.8
9	928	1028	1029.0	1095	56.6	958	1030	1031.6	1095	39.1
10	953	1011	1019.9	1095	47.3	895	1059	1017.1	1074	64.7
11	941	1041	1022.8	1095	50.8	958	1059	1043.7	1095	44.2
12	993	1033	1045.5	1095	30.0	1006	1021	1036.7	1075	26.7
13	955	1060	1039.2	1095	50.0	959	1053	1049.5	1095	38.1
14	946	1006	1012.2	1095	49.0	927	1027	1022.9	1095	59.4
15	949	1021	1019.4	1053	27.0	916	1034	1028.2	1095	56.3
16	891	979	992.2	1075	58.3	995	1068	1054.6	1095	32.5
17	897	1002	1004.4	1095	55.4	958	1054	1045.2	1095	46.4
18	902	952	974.1	1075	53.2	939	963	989.1	1074	47.4
19	969	1025	1033.3	1095	45.5	884	989	990.6	1074	62.6
20	950	1023	1037.6	1095	52.0	985	1027	1039.0	1095	35.3
21	899	1036	1026.9	1095	57.7	940	977	992.1	1075	44.3
22	1016	1032	1043.9	1095	27.2	957	1007	1021.2	1095	49.1
23	947	1036	1027.0	1095	56.6	902	1021	1007.4	1039	38.0
24	977	1068	1053.6	1095	39.7	913	1013	1016.1	1074	48.6
25	945	1021	1031.1	1095	42.9	905	1028	1024.2	1075	53.2
26	1011	1041	1043.7	1095	25.1	952	1031	1036.4	1095	50.1
27	937	1013	1010.0	1095	45.0	961	996	1013.6	1095	49.9
28	971	1002	1027.2	1095	52.1	936	1017	1023.8	1074	46.6
29	907	1038	1027.5	1095	64.6	949	1018	1018.6	1075	32.0
30	950	1018	1016.0	1095	50.4	949	1065	1054.8	1095	41.8

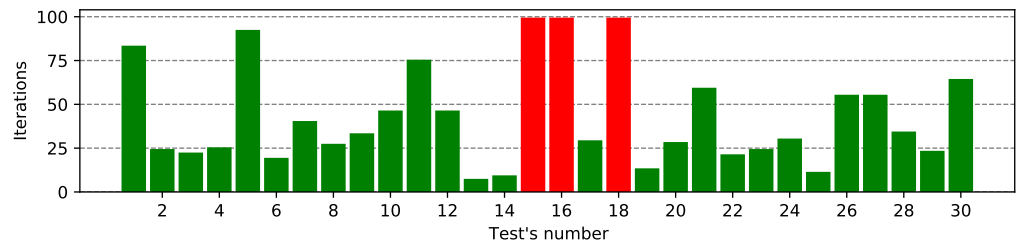


Figure A1. List of correctly guessed bits of MASA attack for the Ω_2 .

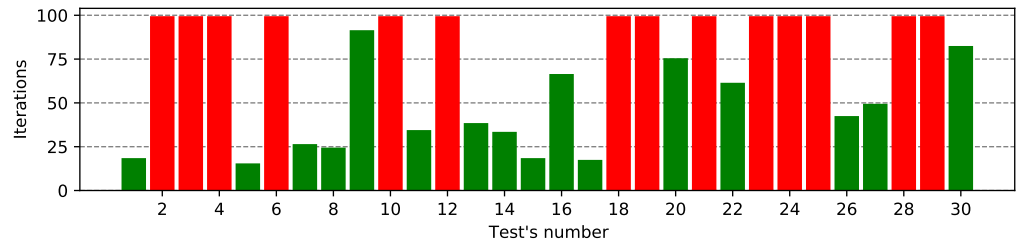


Figure A2. List of correctly guessed bits of NGA attack for the Ω_2 .

Where the red color indicates experiments when the algorithm wasn't able to find the correct subkey and the green bars indicate are tests when the subkey was successfully guessed.

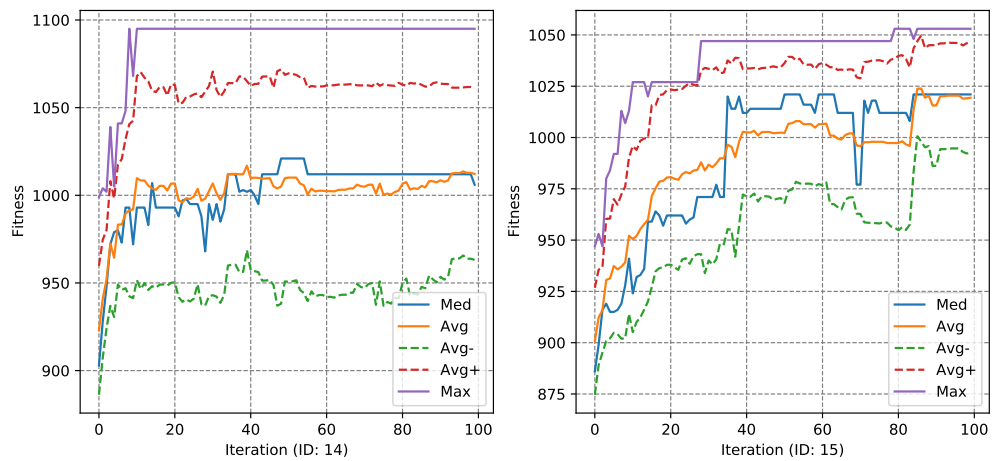


Figure A3. The MASA fitness function F_f convergence diagrams for Ω_2 (tests #14 and #15).

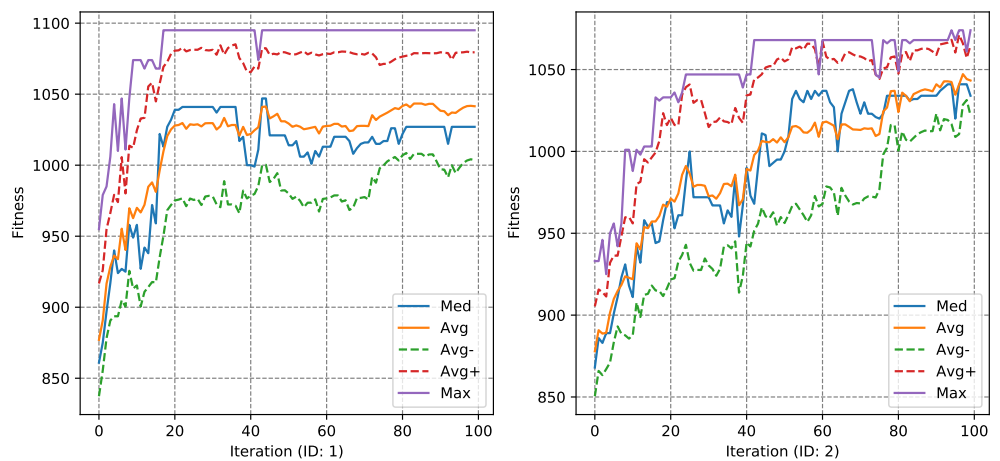


Figure A4. The NGA fitness function F_f convergence diagrams for Ω_2 (tests #1 and #2).

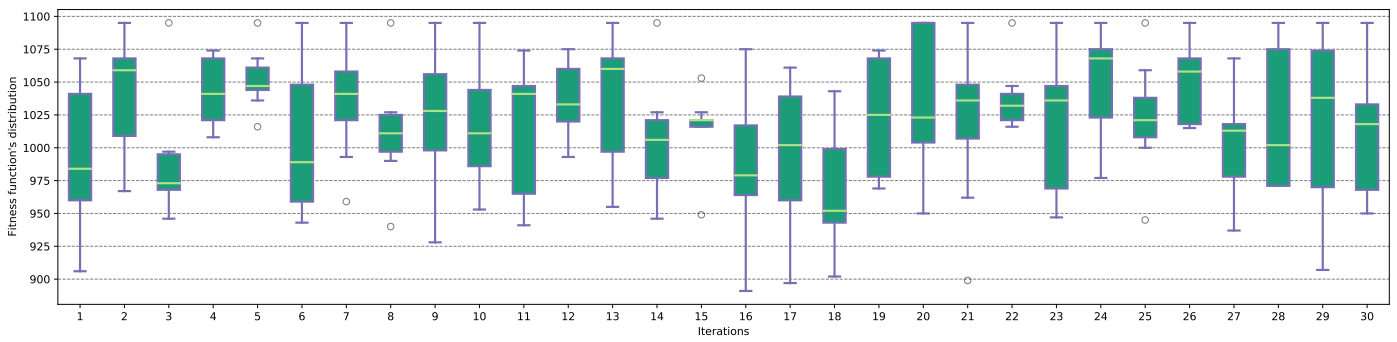


Figure A5. The distribution of the fitness function F_f values in the last iteration for the MASA algorithm and Ω_2 characteristic.

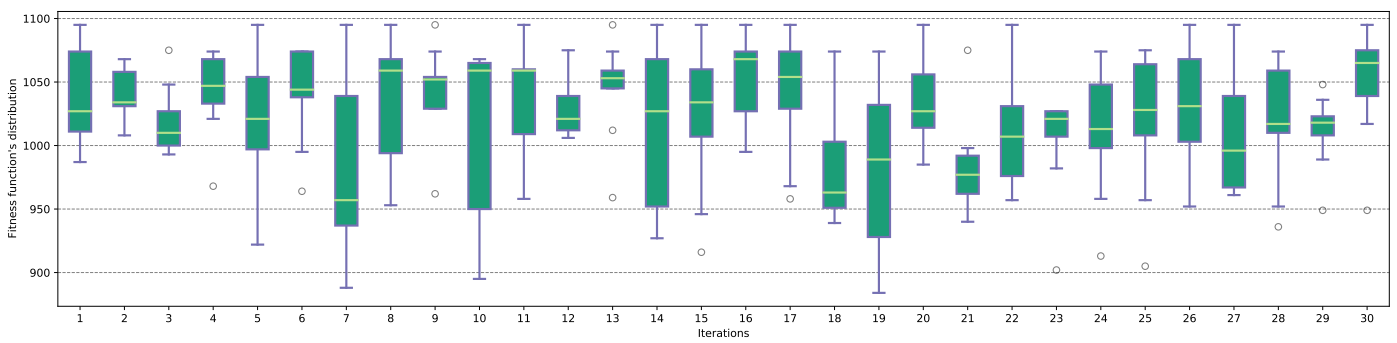


Figure A6. The distribution of the fitness function F_f values in the last iteration for the NGA algorithm and Ω_2 characteristic.

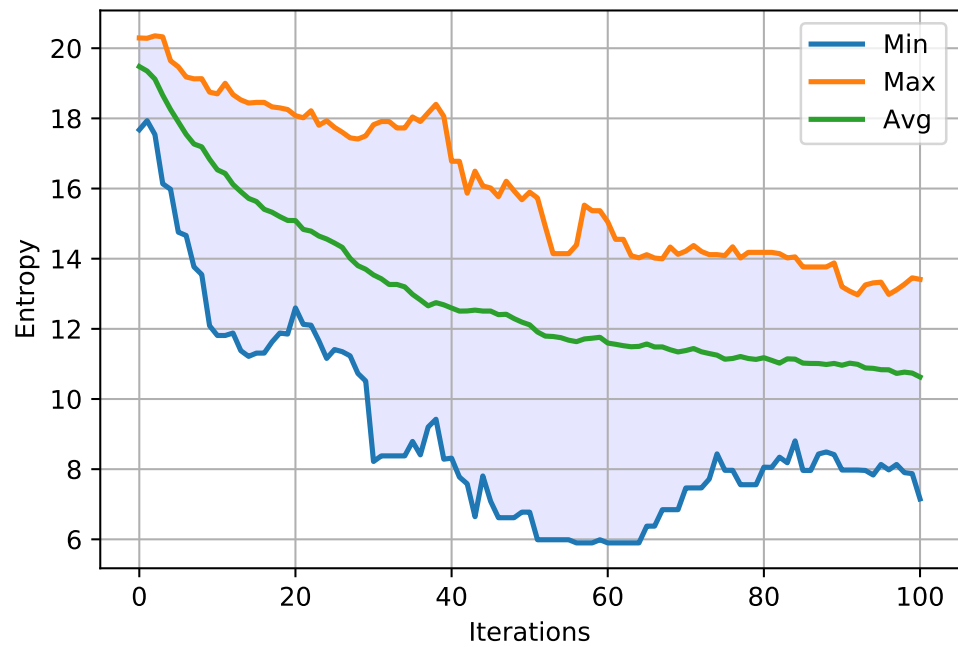


Figure A7. Minimum, maximum and average entropy, during all iterations, for MASA algorithm and Ω_2 characteristic.

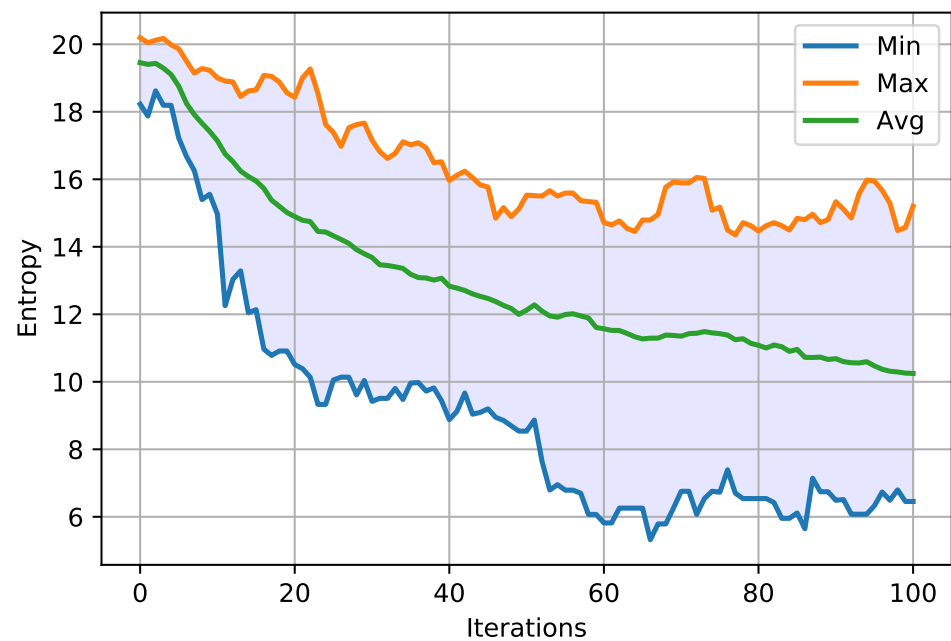


Figure A8. Minimum, maximum and average entropy, during all iterations, for NGA algorithm and Ω_2 characteristic.

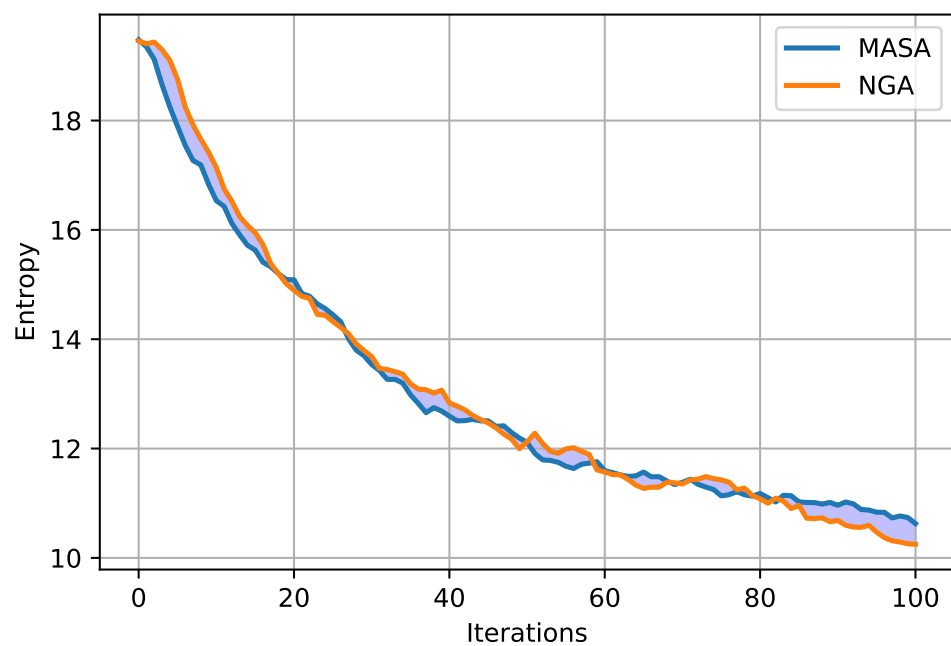


Figure A9. The comparison of the entropy of the MASA and NGA algorithms for the Ω_2 characteristic.

References

- Schneier, B. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*; Wiley: Hoboken, NJ, USA, 1996.
- Menezes, A.J.; Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC Press: Boca Raton, FL, USA, 1997.
- Biham, E.; Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.* **1991**, *4*, 3–72. [[CrossRef](#)]
- Song, J.; Zhang, H.; Meng, Q.; Zhangyi, W. Cryptanalysis of Four-Round DES Based on Genetic Algorithm. *Wirel. Commun. Netw. Mob. Comput. IEEE* **2007**, *10*, 2326–2329.
- Tadros, T.; Hegazy, A.; Badr, A. Genetic Algorithm for DES Cryptanalysis. *Int. J. Comput. Sci. Netw. Secur.* **2007**, *10*, 5–11.
- Garg, P. A Comparison between Memetic algorithm and Genetic algorithm for the cryptanalysis of Simplified Data Encryption Standard algorithm. *Int. J. Netw. Secur. Its Appl. (IJNSA)* **2009**, *1*, 34–42.
- Hu, W. Cryptanalysis of TEA using quantum-inspired genetic algorithms. *J. Softw. Eng. Appl.* **2010**, *3*, 50–57. [[CrossRef](#)]

8. Abd-Elmonim, W.G.; Ghali, N.I.; Hassanien, A.E.; Abraham, A. Known-Plaintext Attack of DES16 Using Particle Swarm Optimization. In Proceedings of the Third IEEE World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 19–21 October 2011; pp. 12–16.
9. Vimalathithan, R.; Valarmathi, M.L. Cryptanalysis of simplified-DES using computational intelligence. *WSEAS Trans. Comput.* **2011**, *10*, 210–219.
10. Jadon, S.S.; Sharma, H.; Kumar, E.; Bansal, J.C. Application of binary particle swarm optimization in cryptanalysis of DES. In *Proceedings of the International Conference on Soft Computing for Problem Solving*; Deep, K., Nagar, A., Pant, M., Bansal, J., Eds.; Advances in Intelligent and Soft Computing; Springer: New Delhi, India, 2012; Volume 130, pp. 1061–1071.
11. Pandey, S.; Mishra, M. Particle swarm optimization in cryptanalysis of DES. *Int. J. Adv. Res. Comput. Eng. Technol.* **2012**, *4*, 379–381.
12. Ali, I.K. Cryptanalysis of simple substitution ciphers using bees algorithm. *J. Baghdad Coll. Econ. Sci. Univ.* **2013**, *36*, 373–382.
13. Boryczka, U.; Dworak, K. Cryptanalysis of Transposition Cipher Using Evolutionary Algorithms. In *Computational Collective Intelligence. Technologies and Applications*; Hwang, D., Jung, J.J., Nguyen, N.T., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; Volume 8733, pp. 623–632.
14. Mekhaznia, T.; Menai, M.E.B. Cryptanalysis of classical ciphers with ant algorithms. *Int. J. Metaheuristics* **2014**, *3*, 175–198. [[CrossRef](#)]
15. Bhateja, A.K.; Bhateja, A.; Chaudhury, S.; Saxena, P.K. Cryptanalysis of vigenere cipher using cuckoo search. *Appl. Soft Comput.* **2015**, *26*, 315–324. [[CrossRef](#)]
16. Jain, A.; Chaudhari, N.S. A New Heuristic Based on the Cuckoo Search for Cryptanalysis of Substitution Ciphers. In Proceedings of the International Conference on Neural Information Processing, Istanbul, Turkey, 9–12 November 2015; Sabri, A., Tingwen, H., Weng, K.L., Qingshan, L., Eds.; Volume 9490, pp. 206–215.
17. Amic, S.; Soyjaudah, K.S.; Mohabeer, H.; Ramsawock, G. Cryptanalysis of DES16 using binary firefly algorithm. In Proceedings of the 2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies, Balaclava, Mauritius, 3–6 August 2016; IEEE: Balaclava, Mauritius, 2016; pp. 94–99.
18. Dworak, K.; Nalepa, J.; Boryczka, U.; Kawulok, M. Cryptanalysis of SDES using genetic and memetic algorithms. In *Recent Developments in Intelligent Information and Database Systems*; Król, D., Madeyski, L., Nguyen, N.T., Eds.; Springer International Publishing: Da Nang, Vietnam, 2016; pp. 3–14.
19. Dworak, K.; Boryczka, U. Differential Cryptanalysis of FEAL4 using Evolutionary Algorithm. In *Computational Collective Intelligence*; Nguyen, N.T., Iliadis, L., Manolopoulos, Y., Trawiński, B., Eds.; Springer International Publishing: Halkidiki, Greece, 2016; Volume 9876, pp. 102–112.
20. Amic, S.; Soyjaudah, K.S.; Ramsawock, G. Dolphin swarm algorithm for cryptanalysis. In *Information Systems Design and Intelligent Applications*; Satapathy, S., Bhateja, V., Somanah, R., Yang, X.S., Senkerik, R., Eds.; Advances in Intelligent Systems and Computing; Springer: Singapore, 2019; Volume 863, pp. 149–163.
21. Jain, A.; Chaudhari, N.S. A novel cuckoo search strategy for automated cryptanalysis: A case study on the reduced complex knapsack cryptosystem. *Int. J. Syst. Assur. Eng. Manag.* **2017**, *9*, 942–961. [[CrossRef](#)]
22. Dworak, K.; Boryczka, U. Genetic Algorithm as Optimization Tool for Differential Cryptanalysis of DES6. In *Computational Collective Intelligence*; Nguyen, N.T., Papadopoulos, G.A., Jędrzejowicz, P., Trawiński, B., Vossen, G., Eds.; Springer International Publishing: Nicosia, Cyprus, 2017; Volume 10449, pp. 107–116.
23. Polak, I.; Boryczka, M. Tabu search against permutation based stream ciphers. *Int. J. Electron. Telecommun.* **2018**, *64*, 137–145.
24. Kamal, R.; Bag, M.; Kule, M. On the cryptanalysis of SDES using binary cuckoo search algorithm. In *Computational Intelligence in Pattern Recognition*; Das, A., Nayak, J., Naik, B., Pati, S., Pelusi, D., Eds.; Advances in Intelligent Systems and Computing; Springer: Singapore, 2019; Volume 999, pp. 23–32.
25. Polak, I.; Boryczka, M. Tabu Search in revealing the internal state of RC4+ cipher. *Appl. Soft Comput.* **2019**, *77*, 509–519. [[CrossRef](#)]
26. Sabonchi, A.K.S.; Akay, B. Cryptanalysis of Polyalphabetic Cipher Using Differential Evolution Algorithm. *Tehnički Vjesnik* **2020**, *27*, 1101–1107.
27. Grari, H.; Lamzabi, S.; Azouaoui, A.; Zine-Dine, K. Cryptanalysis of Merkle-Hellman cipher using ant colony optimization. *IAES Int. J. Artif. Intell.* **2021**, *10*, 490–500. [[CrossRef](#)]
28. Amic, S.; Soyjaudah, K.S.; Ramsawock, G. Binary cat swarm optimization for cryptanalysis. In Proceedings of the 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bhubaneswar, India, 17–20 December 2017; IEEE: Bhubaneswar, India, 2017; pp. 1–6.
29. Pieprzyk, J.; Hardjono, T.; Seberry, J. *Fundamentals of Computer Security*; CRC Press: Boca Raton, FL, USA, 2003.
30. Stallings, W. *Cryptography and Network Security: Principles and Practice*; Pearson: London, UK, 2011.
31. Stinson, D.R. *Cryptography: Theory and Practice*; CRC Press: Boca Raton, FL, USA, 1995.
32. Stamp, M.; Low, R.M. *Applied Cryptanalysis. Breaking Ciphers in the Real World*; Wiley-Interscience: Hoboken, NJ, USA, 2007.