On the Application of Symbolic Regression and Genetic Programming for Cryptanalysis of Symmetric Encryption Algorithm

Tomas Smetka, Ivan Homoliak, Petr Hanacek
Faculty of Information Technology, BUT,
Bozetechova 1/2, 612 66 Brno, Czech Republic
Email: {ismetka, ihomoliak, hanacek}@fit.vutbr.cz

***Abstract*—The aim of the paper is to show different point of view on the problem of cryptanalysis of symmetric encryption algorithms. Our dissimilar approach, compared to the existing methods, lies in the use of the power of evolutionary principles which are in our cryptanalytic system applied with leveraging of the genetic programming (GP) in order to perform known plaintext attack (KPA). Our expected result is to find a program (i.e. function) that models the behavior of a symmetric encryption algorithm DES instantiated by specific key. If such a program would exist, then it could be possible to decipher new messages that have been encrypted by unknown secret key. The GP is employed as the basis of this work. GP is an evolutionary algorithm-based methodology inspired by biological evolution which is capable of creating computer programs solving a corresponding problem. The symbolic regression (SR) method is employed as the application of GP in practical problem. The SR method builds functions from predefined set of terminal blocks in the process of the GP evolution; and these functions approximate a list of input value pairs. The evolution of GP is controlled by a fitness function which evaluates the goal of a corresponding problem. The Hamming distance, a difference between a current individual value and a reference one, is chosen as the fitness function for our cryptanalysis problem. The results of our experiments did not confirmed initial expectation. The number of encryption rounds did not influence the quality of the best individual, however, its quality was influenced by the cardinality of a training set. The elimination of the initial and final permutations had no influence on the quality of the results in the process of evolution. These results showed that our KPA GP solution is not capable of revealing internal structure of the DES algorithm's behavior.**

***Index Terms*—Cryptanalysis, DES, genetic programming, symbolic regression.**

I. Introduction

Standard cryptanalytic methods for symmetric encryption algorithms are based on the principle of finding the key that was used for the confidentiality of the information. They utilize brute force or look for the weaknesses of the encryption algorithms as an attempt of reducing the time to find the key. This work shows that there are other approaches and methods, which can be applied for cryptanalysis. The work leverages evolution which is the drive motor for development of all living organisms on the Earth. The evolutionary mechanisms are applied here using genetic programming, whose purpose is to find a program that models the behavior of a symmetric encryption algorithm. If such a model would be constructed, then it could be able to decipher all new messages which have been encrypted by modeled symmetric encryption algorithm.

The paper describes the basic principles of genetic programming with symbolic regression, and subsequently presents the cryptanalytic system which utilizes the mentioned principles. The main part of the work is devoted to the cryptanalysis of the DES algorithm by using of genetic programming. Later, the paper summarizes the results and lists the possible extension of the work.

II. Algorithm DES

Data Encryption Standard (DES) [1] is chosen as an analyzed symmetric encryption algorithm because of its standardization and known weaknesses. The IBM company is standing behaind the development of the DES [1]. DES encryption algorithm is an enhanced version of the Lucifer [2] algorithm. DES represents type of Feistel cipher, that is utilized as block symmetric encryption algorithm.

At the time of introducing DES as a standard, there were some concerns and doubts about its safety – particularly by Hellman and Diffie [3], [4]. Later, Morris, Sloane and Wyner published two possible weaknesses of DES [5], which are based on: a) the key length of 56 bits and thus may not provide adequate security, and b) the S-boxes, which may contain hidden backdoor.

III. Genetic Programming

Genetic programming (GP) represents biologically inspired methods which are able to create computer programs that solve the high-level problems [6]. In GP, the problem of artificial intelligence, machine learning, adaptive systems and automatic learning is transformed into the searching for a computer program. GP provides us with a way to find a computer program that resolves the problem in the area of computer programs [7].

The essential difference between the evolution in sciences and genetic programming is that evolution is in progress in the sciences by permanently changing the environment and without the objective (no final state), while the objective of the evolution in genetic programming is defined by fitness functions [8], [9].

Genetic programming does not have limited structure (fixed length of the chromosome) when searching for a suitable solution of the given problem [10]. Individuals (programs) of GP can potentially take unlimited complexity [11].

A. Language of the Representation

The code of each individual is formed by using a set of functions and a set of terminals in GP [12], [13]. Programs in GP work with so-called executable structures. The most commonly used structures are syntactic trees, because they are suitable for machine processing and can be used in almost any programming language [7].

B. Set of Terminals

Inputs of the programs, that are evolved using GP, are represented by set of terminals. This set contains constants, nullary functions (a function without arguments) and variables.

C. Set of Functions

Set of utilized functions is dependent on the area of the problem which is being solved by GP. The next parts of the set can be arithmetic and logical operations, classic constructs of the common programming languages such as conditions, cycles, and subroutines. It is not desirable to use very large set of functions, because it increases the search space, and the solution may not be found then [9].

D. Fitness Function

In order to determine which programs or parts of the searching space are feasible (the ones which address or approximate the solution of our problem) is used the fitness function. Fitness function expresses the quality of the individual and is a key mechanism of the navigation in that searching space, and thus of the convergence to an acceptable outcome [8].

The fitness function is determined by the programmer in advance and it is not the subject to the process of the evolution. The quality measurement of an individual can be done in many ways, while the particular form of fitness function depends on specification of the problem. The basic approaches to the measurement of the quality of the individuals are semantic analysis and consensus on the training set [10]. The terms for the fitness values such as gross fitness, standardized fitness or normalized fitness can be found in the literature [7].

E. Generation of the Initial Population

The first step of the GP algorithm is to initialize the initial population, which is generated randomly [8] The initialization process of the population randomly selects the symbols from a set of terminals and a set of functions, and then it creates the programs represented by the tree structure according to the specified rules. Generation of the initial population can be accomplished using a variety of methods, such as *Grow Full* and *Ramped Half and Half*.

F. Genetic Operators

In genetic programming are defined the following basic operators: crossover, reproduction and mutation [12], [14]. New population is created from the selected individuals by application of genetic operators in terms of evolutionary process [14]. Therefore, the initial population is transformed using the genetic operators to final one during the whole evolutionary process.

1) Reproduction: Is the operator, which according to the established selection mechanism chooses the appropriate individuals for reproduction into the new generation [14]. The selective mechanism mimics natural selection according to the Charles Darwin's theory: better adapted individuals are more likely to reproduce. This mechanism ensures that the average quality of the population increases with the number of generations.

The selection of individuals for reproduction into the next generation has to sufficiently prioritize high-quality individuals (having higher fitness), however it also has to create new generation sufficiently varied. If the reproduction does not meet any of the two mentioned requirements, then it may lead to premature convergence while high quality individuals are preferred, and thus local solution may be found. However, in the second case, it may lead to the slow convergence of the algorithm [9]. The preference level of quality individuals and the suppression of the low-quality individuals is a property of a selection mechanism, which is called *selection pressure*. There exist plenty of selection operators for choosing appropriate individuals into the next generation. The best known are proportional selection (roulette wheel selection) and tournament selection [15].

2) Crossover: The crossover operator is working with two individuals (parents) and combines their genetic material by selecting a part of one parent with a part from the second parent. This creates two new individuals (offspring), who are placed into the new generation of the evolution process [7].

The basic variant of crossover is called *crossover of sub-tree* and is applied as follows. The selection method chooses two individuals as parents. The point of crossover (node) is determined in each parent. The sub-trees, which are located under the crossover points are swapped between the two parents. The result are two new individuals, who are propagated into the new generation.

3) Mutation: The mutation operator is working with one individual, and as the only one is able to introduce the information into the system. Usually, the mutation is random and its probability is very low. Koza has demonstrated that the mutation is not necessary ($P_m = 0$) [7] or is recommended to be very small [12], e.g. $P_m = 0.05$. The most commonly used type of mutation is called *mutation of sub-tree*. It is randomly selected a node in the tree of an individual (mutation point) and its sub-tree is replaced by the newly generated tree.

G. End of the Evolution

Evolution in the sciences is represented as a never ending process. However, it is not desired property when solving

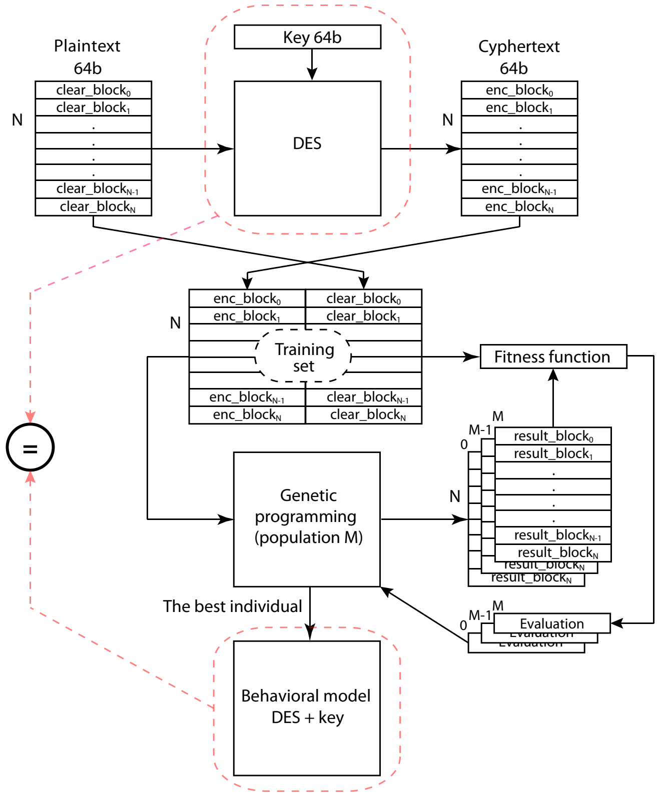


Fig. 1. Behavior of the DES algorithm using genetic programming

the real problem using the power of evolution. Therefore, we use different criteria in order to stop the evolutionary process of genetic programming. The most commonly used criteria are the number of generations, the computing time, achievement of the desired fitness value or the convergence of the population [16].

IV. Symbolic Regression

Genetic programming can be in principle applied to many areas. Nevertheless, the goal of many problems is to find a function that has some required properties, e.g. the function values correspond to the target values for specified input values. This procedure is generally known as *symbolic regression* problem, which is one of the first applications of genetic programming in practice [7].

Symbolic regression is based on finding such a function of the program, which creates the required output values of the specified input values without any assumptions about the structure of this function [10], [7]. Genetic programming is optimal for this type of discovery, as it does not consider such assumptions.

V. Cryptanalysis Using Genetic Programming

By combining the genetic programming and symbolic regression, it was designed cryptanalytic system, which is depicted in Fig. 1. The principle of proposed cryptanalytic system is the following:

1) **Generating of a training set** – cryptanalytic system works with the Known Plaintext Attack (KPA). Therefore, it is necessary to know the pair of plain text and the corresponding cypher text ($64b$ blocks in the DES case). Sufficient size of a training set could be $N = 1000$ of such pairs. The upper part of Fig. 1 contains example of generating of a training set.
2) **Genetic programming** – using evolutionary principles, there are constructed programs based on data from the training set. The aim of the program interpretation is to get output data, which are the same as the reference data. There is used the fitness function in order to compare the similarity of both data values and thus determine the quality of an individual.
3) **Fitness function** – its basis is the function $hDist$ which represents the distance of the two $64b$ strings: A (reference block) and B (block obtained from the interpretation of program), where the index of i (starting from 1) represents the position of the bit in the string. Therefore, it is the sum of the bit differences between the two data blocks. The resulting $fitness$ function is the average of Hamming distance on the whole training set. The following equations defines fitness function:

$$hDist = \frac{1}{64}\sum_{i=1}^{64} A_i \oplus B_i \qquad (1)$$

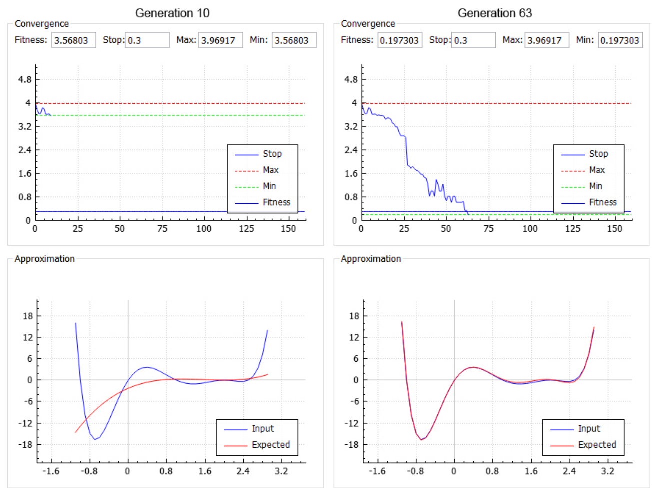
$$fitness = \frac{1}{N}\sum_{i=1}^{N} hDist_i \qquad (2)$$

4) **The best individual** – models the behavior of the selected encryption algorithm parametrized with the key, which was utilized for generation of the training set. Such an individual should be able to decipher other encrypted blocks than the ones from the training set.

A. Functionality Validation of Genetic Programming

The implementation of the simulation tool is a part of this work. This tool is working with genetic programming under the proposed cryptanalysis system (see Fig. 1) and allows application of a symbolic regression method on an input data without focus on the DES algorithm. The functionality of all parts of the genetic programming is checked on a set of validation tests.

Validation is performed on a set of mathematical functions involving different degrees polynomials and trigonometric functions. The nodes `IF` and `FOR` are validated on the training set generated by a factorial function. Syntax tree of the best found individual is shown in Fig. 3. Measured results of one of the validation tests, in which training set is generated by polynomial $2x^6 - 13x^5 + 26x^4 - 7x^3 - 28x^2 + 20x$, are presented in Fig. 2. These results show corresponding graph of convergence and approximation graph for the chosen generations. The approximation graph shows reference values from the training sets in the blue color and red color represents



Fitness: 3.56803 Stop: 0.3 Max: 3.96917 Min: 3.56803

Fitness: 0.197303 Stop: 0.3 Max: 3.96917 Min: 0.197303

Fig. 2. Fitness convergence and approximation graph for chosen generations

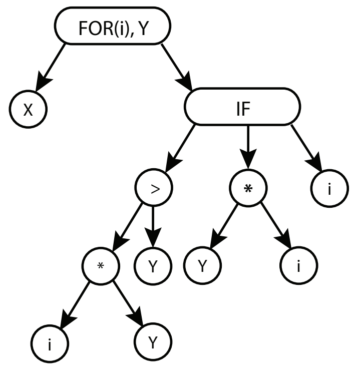


Fig. 3. Syntactic tree of the evolved individual

the output of the function/program of the best individual in the chosen generation.

VI. Testing and Evaluation of Achieved Results

Symbolic regression method works with a training set of pairs of an opened text P and an encrypted text C that is generated by DES algorithm. The pairs of a training set are represented by `uint_64` data type. Experiments are focused on diversely large training sets generated by different number of rounds of DES algorithm. The goal of the experiments is to construct, using the evolution process such an individual, that simulates the expected behavior of DES algorithm for a corresponding training set.

A. Fitness Value Estimate

The quality of an individual is directly proportional to its fitness value. Fitness function heuristics is determined by the average Hamming's distance, thus a bit difference average between the output of the individual and the corresponding reference value from a training set. What fitness values can an individual get?

Theorem 1: Each bit of randomly generated data block can have values 1 or 0 with probability of $p = 0.5$. Let's take 2 different randomly generated data blocks X and Y with the same number of bits N. Their bit difference complies with the following

$$count(X \oplus Y) \approx N * (1 - p), \tag{3}$$

where *count* is a function counting number of occurrence of bits with value 1.

Theorem 2: Taking into account an encryption algorithm processing blocks of size N, there should be no relation between the opened block P and the corresponding block C. We can assume that data blocks are random in relation to each other. Coming out from Theorem 1, Equation 3 must be true, thus the bit difference of P and C corresponds to the expression $N * (1 - p)$, where $p = 0.5$.

Corollary 1: DES algorithm processes blocks of size $N = 64$ bits. Applying Equation 3 to the block size N, it should give us that the average bit difference between blocks of opened text P and encrypted text C is ≈ 32 bits.

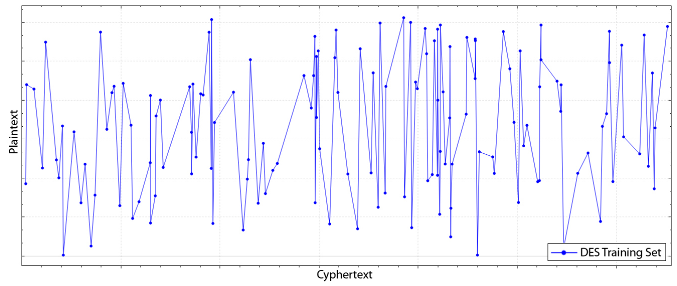


Fig. 4. Visualization of training on DES

B. Visualization of Training Set of DES Algorithm

Imagine DES algorithm in decryption mode as a function which transforms the encrypted input block C into the opened block P. Blocks of text are represented by data type `uint_64` and each block is symbolized by the corresponding decimal value. A course of the function of DES algorithm in decryption mode is present in Fig. 4, which shows values for 200 pairs of P and corresponding C from the training set generated by 8 DES rounds with a randomly selected secret key.

Although the figure does not constitute the optimal representation for the selected heuristics of the fitness function, emerging from the visual comparison with i.e. courses of polynomial functions it can be assumed that finding a fine individual is not a trivial task. It can be also visually verified that there is no obvious relation between pairs of blocks P and C from the training set. The course of the DES function corresponds to the course of randomly generated data.

The data shown in Fig. 4 are given in a text format to the genetic programming process which works with the symbolic regression method. Using the data and evolutionary principles, it tries to generate individuals whose program representation transforms C into P, thereby simulates the process of decryption of DES algorithm.

C. Practical Confirmation of the Assumptions

Theorem 2 is practically confirmed by using a simple iterative algorithm, whose pseudo code is described in Algorithm 1. The input of Algorithm 1 is a random secret key for encryption by the DES algorithm and the number of iterations for the calculation. With the increasing number of iterations, statistical accuracy is increased too – in our case it was used 2^{20} iterations. Considering Theorem 2, the expected result for DES working with blocks of 64 bits in size is $\approx$ 32 bits, which is confirmed by the algorithm.

Algorithm 1: Pseudo code of the bit difference between P and C

```
Function convergence_assumption(attempts, key) :double
    int diff = 0;
    for (i = 0; i < attempts; i++) do
        plain = generate_random_plain();
        cypher = des_encrypt(plain, key);
        diff += count(plain ⊕ cypher);
    return diff/attempts;
```

Corollary 2: The output of each individual within the evolutionary process is a data block with the size of the 64 bits. If we leverage Corollary 1 related to the lowest accuracy of the GP – GP does not find a connection on the training set and the outputs of the individuals are random) – then the maximal difference between an individual's output and the reference block is $\approx$ 32 bits. Also, this value represents the worst theoretical evaluation of the fitness of the individual. Therefore, we assume for the experiments that if the evolution converges, then the fitness value is improving from the worst possible values of 32 bits.

D. Finding of the Best Parameters

We need to specify the parameters of GP before the experiments themselves. Finding the optimal parameters of GP requires a good understanding of the solved problem. In the current work, we focus on the DES algorithm, which is implemented using cycles, rotations and the bitwise XOR operations. Therefore, these operations are included into the set GP functions. Subsequently, we experiment with various training sets, different sets of functions, terminals and various settings of the genetic programming. The main indicator during the search for optimal parameters is the slope of convergence evolution chart. The result of the experiment is to find the best parameters of GP, which are described in Table I.

E. Initial and Final Permutation

By the following experiments, we check whether the initial and final permutation has any influence on the results of the fitness values. Each block C of the training set is the result of the encryption process of DES algorithm with input block P. In the first step, the input block P is transformed by the initial permutation IP. Then, a defined number of encryption rounds is executed, and finally the output is transformed by permutation IP^{-1}. By the operation of removing the initial and the final permutation, we understand the operation that transforms the input training set according to Equation 4.

$$\begin{aligned} input &= rev\_IP^{-1}(cyphertext) \\ output &= IP(plaintext). \end{aligned} \quad (4)$$

Theorem 3: Initial and final permutations are computational operations that burden the evolutionary process of GP by the need of revealing the more dependences of the DES algorithm. If these redundant computational operations are removed from the training set, then we expect to achieve better results by newly constructed individuals compared to the original training set.

Methodology of the Measurement: The measurement takes place on the training sets of size $N = 100$. The training set of size N will be separately generated by the DES algorithm with the number of rounds $R \in \{2, 4, 8, 16\}$. 50 measurements are done for each such training set and it records the value of the fitness F at 50th generation ($g = 50$). Table II contains the arithmetic mean of the F values. Also, the experiment is performed again, while this time we remove the initial and final permutation, which is depicted by column called *Removed*. Theorem 3 is not confirmed by the results of the fitness values measured in the training sets that did not contain initial and final permutations versus the original training sets.

TABLE I
THE OPTIMUM PARAMETERS OF THE DES CRYPTANALYSIS USING GP

Parameter	**Attribute**
Terminal set	T = {x, random constants $\{0, 1, ..., 64\}$}
Function set	F = {$+$, $-$, SL, SR, RL, RR, AND OR, XOR, FOR max. 64 of cycles}
Population size	4000 individuals, unlimited size of tree
Population initialization	Ramped Half-and-Half (depth 2-3, 50% of terminals are constants)
Selection method	Tournament selection (selection pressure = 5)
Fitness heuristics	Average Hamming distance
Genetic operators	80% crossover, 10% reproduction, 10% mutation
Crossover parameters	50% crossover of the highest sub-tree, 50% crossover of the random sub-tree
Mutation parameters	45% mutation of terminals, 45% mutation of functions, 10% mutations of sub-tree, generation of sub-tree by Ramped Half-and-Half method (depth 2-3, 50% of terminals are constants)

TABLE II
DEPENDENCY OF THE PERMUTATION ON THE FITNESS VALUE

DES Rounds	**Permutation**	
	Included	**Removed**
2	28.47	28.40
4	28.51	28.54
8	28.54	28.52
16	28.43	28.38

Although the transformation of the training set by removing the initial and final permutations has not produced better results, we decided to include this operation into the set of the best parameters of GP. Therefore, all the next experiments assume the use of the optimal parameters from Table I as well as removing of IP and IP^{-1} from the training sets.

F. Evolution of the Best Individual

Theorem 4: The encryption strength of the DES algorithm grows with the number of rounds that are applied during encryption. It should be computationally easier for GP to reveal the context of DES algorithm on the training set generated with a lower number of encryption rounds. Thus, the quality of an achieved fitness value should be directly proportional to the number of encryption rounds.

Theorem 5: If the GP could be able to reveal the context of the DES algorithm, then the size of the training set should not affect the results of fitness values. Therefore, the fitness value of the found solution should be similar for different-sized training sets that have been generated by DES with the same number of encryption rounds.

Methodology for Measurement: The measurement is performed on training sets of size $N \in \{10, 100, 1000\}$. Each training set is independently generated by the DES algorithm considering the number of laps $R \in \{2, 4, 8, 16\}$. And each such training set is made of 50 measurements. During each measurement, we let the evolution converge into a state where the fitness reaches the steady value. The fitness value is regarded as a steady, if the following relationship is true:

$$\left| \left(\frac{1}{10} \sum_{i=1}^{10} fitness_{(g-i)} \right) - fitness_g \right| < 0.02, \qquad (5)$$

where $fitness$ represents the value of the best solution in the generation g.

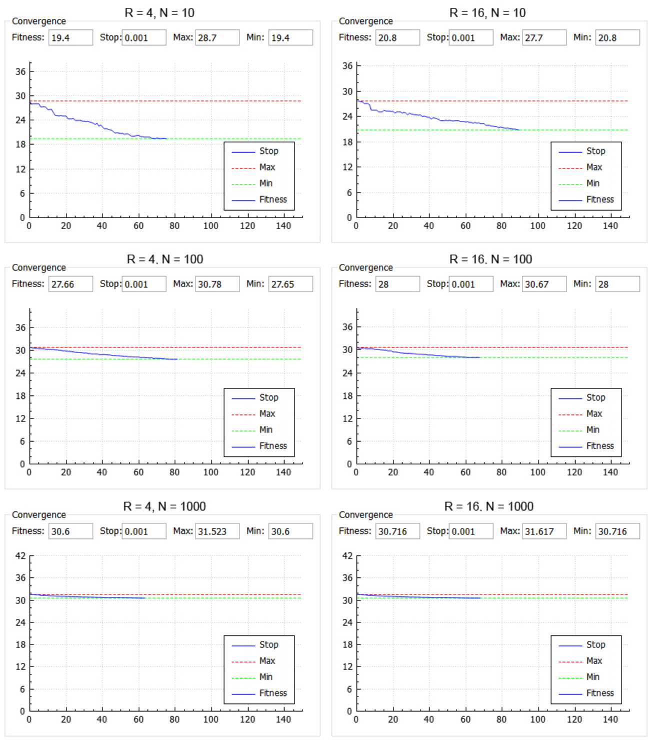
Considering the number of measurements, we calculate the arithmetic means from the obtained results – see Table III. The values in the table represent the average of the best achieved $fitness$ values. For example, consider $N = 10$ and $R = 8$, which have associated value of 19.5. It means that for given training set, GP can transform any C to P, that differs from the reference plain-text in 19.5 bits (of 64 possible). Therefore, an individual is successful at $\approx 70\%$ in decryption of chosen training sets. The examples of the convergence of the best fitness values from the set N, considering R encryption rounds, are shown in Fig. 5.

None of the assumption denoted by Theorem 4 and Theorem 5 has been confirmed by the results of the experiments. The accuracy rate of the fitness value convergence does not depend on the number of DES encryption rounds, which is utilized in generation of the training set.

It can be observed that with the growing number of training pairs N, the best fitness value is exacerbated. The measurement results indicate that genetic programming works successfully only within the input training sets and is not able to detect the dependencies in the DES algorithm.

TABLE III
AVERAGE FITNESS VALUES OF THE BEST EVOLVED INDIVIDUALS

DES rounds	**Number of pairs in a training set**		
	10	**100**	**1000**
2	19.34	27.85	30.81
4	19.41	28.01	30.72
8	19.50	27.79	30.77
16	19.33	27.83	30.73



R = 4, N = 10 — Fitness: 19.4 Stop: 0.001 Max: 28.7 Min: 19.4
R = 16, N = 10 — Fitness: 20.8 Stop: 0.001 Max: 27.7 Min: 20.8
R = 4, N = 100 — Fitness: 27.66 Stop: 0.001 Max: 30.78 Min: 27.65
R = 16, N = 100 — Fitness: 28 Stop: 0.001 Max: 30.67 Min: 28
R = 4, N = 1000 — Fitness: 30.6 Stop: 0.001 Max: 31.523 Min: 30.6
R = 16, N = 1000 — Fitness: 30.716 Stop: 0.001 Max: 31.617 Min: 30.716

Fig. 5. Convergence of fitness value for $R \in \{4, 16\}$, $N \in \{10, 100, 1000\}$

G. Generalization of the Model

The following experiments describe how the best individuals model expected solution and how can generalize.

Theorem 6: Every individual is adapting the training set of pairs P_{ref} and C_{ref} during the evolution process, and on the basis of its quality is computed adequate $fitness$ according to the relation

$$count(individual(C_{ref}) \oplus P_{ref}) = fitness, \qquad (6)$$

where *individual* represents the current individual and *count* represents the function, that counts the number of bits set to 1.

Theorem 7: Now suppose the same individual *individual* from the provided 6, but with the difference that P is a randomly generated open block, C is the corresponding encrypted block using the same secret key and the number of rounds, as in the training containing pairs P_{ref} and C_{ref}. We use the same relation for the computation of the fitness value

$$count(individual(C) \oplus P) = fitness_g. \qquad (7)$$

If $32 \gg fitness_g \geq fitness$, then the individual can generalize the final solution. If $fitness_g = fitness$, then the individual represents the optimal solution in the scope of achieved $fitness$.

Theorem 8: If $fitness_g \approx 32$ is true, then we can assume that constructed individual does not generalize the final solution.
Iterative algorithm was designed in order to verify generalization, which is the part of the simulator of the genetic programming. The activity of the algorithm is described by pseudo code in Algorithm 2.

The input of the algorithm is the secret key, the number of rounds of the DES algorithm and the number of iteration of the algorithm. Note that statistical precision of the result increases with the increasing number of the iterations. Value of 2^{20} iterations is estimated as sufficient for the experiment.

Algorithm 2: Pseudo code of algorithm validating generalization

```
Function generalization(attempts, individual, key, rounds) :double
    solution = 0;
    for (i = 0; i < attempts; i++) do
        plain = generate_plain();
        cypher = des_encrypt(plain, key, rounds);
        plain = IP(plain);
        cypher = rev_IP$^{-1}$(cypher);
        out = individual$\rightarrow$evaluate(cypher);
        solution += count(out $\oplus$ plain);
    return solution/attempts;
```

Methodology of the Measurement: The measurement takes place on the training sets of size $N \in \{10, 100, 1000\}$, where the results for each N represents Table IV, Table V and Table VI. Each training set of size N is independently generated by the DES algorithm having the number of laps $R \in \{2, 4, 8, 16\}$. For each training set is performed 50 measurements, and during the evolution, generalization value G is calculated in the generations $g \in \{0, 25, 50, max\}$ as well as achieved fitness value F is recorded. Generation max represents the generation in which it is reached a steady fitness value. Mentioned tables contain arithmetic average of the values F and G, computed from the all experiments.

The results of the measurement show that regardless of the size of the training set, the number of encryption rounds and the generation in which the fitness value has reached interesting level of generalization, the generalization corresponds to the difference of ≈ 32 bits.

H. Overview of Measured Results

Results of generalization values confirm Theorem 8, thus the best individuals do not generalize the searched solution. The method of symbolic regression has shown itself as successful in terms of convergence of the searched solution on the provided training set. However, the best individuals do not represent DES algorithm in decryption mode in any aspect. The behavior of the best found individual is similar to, based on Theorem 1, a program behavior which generates random data blocks in comparison with the expected outputs. The measured results of unsuccessful generalization might be caused by several factors.

- **Parameters of genetic programming** don't have to represent optimal parameters. On that account we are unable to decide whether a solution is sufficiently good or whether the best solution is found.
- **Scalability problem** is one of the common problems of GP and it states that current simple programs created by evolution process contain hundreds of nodes. Despite the syntax trees of experimental individuals contained up to 6000 nodes, they reached a maximum depth of 70 and thus this program representation might not have been sufficient to cover the required program capabilities of the searched solution.

TABLE IV
GENERALIZATION RESULTS FOR $N = 10$

DES Rounds	Generation							
	0		25		50		Max	
	F	G	F	G	F	G	F	G
2	28.03	32.004	23.86	32.001	21.42	32.002	19.34	31.998
4	28.20	32.000	23.74	32.002	21.47	31.996	19.41	32.001
8	28.13	32.001	24.02	31.999	21.46	31.999	19.50	32.003
16	27.96	31.999	24.05	31.996	21.56	31.998	19.33	32.000

TABLE V
GENERALIZATION RESULTS FOR $N = 100$

DES Rounds	Generation							
	0		25		50		Max	
	F	G	F	G	F	G	F	G
2	30.67	31.999	29.50	32.000	28.46	32.000	27.85	32.002
4	30.78	32.002	29.48	32.001	28.50	32.000	28.01	31.997
8	30.62	31.998	29.42	31.996	28.58	31.999	27.79	32.001
16	30.71	32.003	29.37	32.000	28.42	32.002	27.83	31.999

TABLE VI
GENERALIZATION RESULTS FOR $N = 1000$

DES Rounds	Generation							
	0		25		50		Max	
	F	G	F	G	F	G	F	G
2	31.63	31.998	31.20	31.999	31.03	32.000	30.81	31.999
4	31.62	32.001	31.17	32.002	30.98	31.999	30.72	31.997
8	31.58	31.999	31.15	31.998	30.95	32.001	30.77	32.000
16	31.64	32.000	31.14	32.000	30.97	32.001	30.73	31.999

- **Complexity of DES algorithm** is enormous in comparison with tasks currently solved by GP. The computation of DES algorithm is parameterized with a 2^{56} bit key and they both imply a large program scope in which it is highly unlikely to find a correspodning model taking into account current technical options.

VII. Conclusion

In our work we designed and implemented the utilization of genetic programming and symbolic regression in a field of cryptanalysis of symmetric encryption algorithms. The attack is performed on well known representative of symmetric encryption algorithms, – DES. Our expected result was to find a program (i.e. function) that models the behavior of a symmetric encryption algorithm DES instantiated by specific key. If such a program would exist, then it could be possible to decipher new messages that have been encrypted by unknown secret key. In the result, we did not confirm the expectation.

Possible further development of our study may be focused on experimentation with syntactic tree structure. Functional and terminal sets may be extended by other types of nodes. Syntactic trees of an individuals can work with automatically defined functions. Other types of fitness functions like N-gram or sub-sequence kernel can be used in further experiments.

From the results of the measurements related to the complexity of the DES algorithm and the problem of scalability, we do not assume that, considering current technical capabilities, it might be constructed solution that would be applicable for successful decryption in practice. If we consider the operation of Moore's rule, then GP will be able to search for the larger program space with the increasing computing performance. But on the other hand, the increased computing performance will reflects the increased security of new encryption algorithms, which will imply searching in more extensive program space needed to perform cryptanalysis by GP.

Acknowledgement

This article was created within the project Reliability and Security in IT (FIT-S-14-2486) and supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science – LQ1602.

References

[1] *Data Encryption Standard*, ser. FIPS pub. 46, National Bureau of Standards. U.S. Department of Commerce, Jan. 1977.

[2] H. Feistel, "Cryptography and computer privacy," *Scientific american*, vol. 228, pp. 15–23, 1973.

[3] M. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, and S. Pohlig, *Results of an Initial Attempt to Cryptanalyze the NBS Data Encryption Standard.* Information Systems Lab., Dept. of Electrical Eng., Stanford Univ., 1976.

[4] W. Diffie and M. E. Hellman, "Special feature exhaustive cryptanalysis of the nbs data encryption standard," *Computer*, vol. 10, no. 6, pp. 74–84, Jun. 1977.

[5] R. Morris, N. J. A. Sloane, and A. D. Wyner, "Assessment of the nbs proposed federal data encryption standard," *Cryptologia*, vol. 1, no. 3, pp. 281–291, 1977.

[6] M. Affenzeller, S. M. Winkler, S. W. 0002, and A. Beham, *Genetic Algorithms and Genetic Programming - Modern Concepts and Practical Applications.* CRC Press, 2009.

[7] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA, USA: MIT Press, 1992.

[8] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming.* Lulu Enterprises, UK Ltd, 2008.

[9] J. Schwarz and L. Sekanina, *Applicable genetic algorithms*, 1st ed., Faculty of Information Technology, Brno University of Technology, 2006, [in czech language].

[10] M. Dostal, *Evolutionary computing techniques*, Department of Computer Science, Palacky University Olomouc, Olomouc, 2007, [in czech language]. [Online]. Available: http://phoenix.inf.upol.cz/esf/ucebni/evt.pdf

[11] T. Weise, *Global Optimization Algorithms - Theory and Application*, 2nd ed. Self-Published, 2009. [Online]. Available: http://www.it-weise.de/

[12] W. Banzhaf, F. D. Francone, R. E. Keller, and P. Nordin, *Genetic Programming: An Introduction: on the Automatic Evolution of Computer Programs and Its Applications.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998.

[13] J. Hynek, "Genetic algorithms in a nutshell," *In Economics and Management*, vol. 5, pp. 48–54, 2002.

[14] ——, *Genetic algoritms and genetic programming.* Praha: Grada Publishing a.s., 2008, [in czech language].

[15] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA, 1989, no. 2.

[16] M. Safe, J. Carballido, I. Ponzoni, and N. Brignole, "On stopping criteria for genetic algorithms," *Advances in Artificial Intelligence–SBIA 2004*, pp. 405–413, 2004.