Combinatorial Optimization in Cryptography

***Abstract*—The known attacks on different cryptosystems lead to a number of criteria that the implemented cryptographic algorithms (ciphers) must satisfy. The design of cryptographic systems needs to consider various characteristics simultaneously, which can be regarded as a multi-objective combinatorial optimization problem. Evolutionary computation present a range of problem-solving techniques based on the principles of biological evolution. Evolutionary algorithms can quickly offer satisfactory solution to combinatorial optimization problems. Evolutionary computation can be also used in evolving pseudorandom number generators which play important role as a countermeasure against side channel attacks. The purpose of this paper is to give a state-of-the-art overview of the evolutionary computation area in symmetric and asymmetric cryptography, as well as for the evolving pseudorandom number generators. In symmetric cryptosystem, one of the important components is the substitution box which can be successfully built by evolutionary algorithm. In asymmetric cryptosystem, evolutionary algorithms can be used to speed-up some discrete mathematic operations, like modular exponentiation.**

I. Introduction

Combinatorial optimization is a topic that consists of finding an optimal object from a finite set of objects. In many such problems, exhaustive search is not feasible. It operates on the domain of those optimization problems, in which the set of feasible solutions is discrete or can be reduced to discrete, and in which the goal is to find the best solution.

Cryptography is the study of mathematical techniques for all aspects of information security. The security of information encompasses the fundamental aspects: confidentiality, data integrity, authentication, and non-repudiation. To ensure all the previous aspects, cryptographic algorithms and modes should satisfy a number of criteria, which can be regarded as a multi-objective combinatorial optimization problem.

Evolutionary computation algorithms represent a range of problem-solving techniques based on principles of biological evolution. Such algorithms can be used to solve a variety of difficult problems, among which are those from area of cryptography and which can be represented as combinatorial optimization problem. It is justified to mention several other methods like cellular automata or addition chains, which in cooperation with evolutionary algorithms can result with better results than conventional algorithms.

This paper is not intended as a detailed survey of possible applications of evolutionary algorithms in cryptography nor detailed explanation of mathematical background in cryptography. Rather, the purpose of this paper is to give a state-of-the-art overview of evolutionary computation methods applied to cryptographic systems design.

The paper begins by discussing evolutionary computation algorithms in Section II. Section III describes relevant theory in cryptography and the criteria that cryptographic algorithms should satisfy. In Section IV the state-of-the-art of different cryptographic problems and the applications of evolutionary computation methods used in cryptography is given. Finally, Section V draws a conclusion.

II. Evolutionary Computation

Evolutionary computation is a family of algorithms inspired by biological evolution, and the subfield of artificial intelligence and soft computing. It can be roughly divided to three groups: evolutionary algorithms, swarm algorithms, and other algorithms. Evolutionary computation uses iterative progress, like growth or development in population. This population is then selected in a guided random search to achieve the desired goal. In following paragraphs, briefly is explained each of these three groups of evolutionary computation.

A. Evolutionary Algorithms

Evolutionary algorithms could be broadly divided into 4 different approaches: genetic algorithms [1], genetic programming [2], evolutionary strategies [3], and evolutionary programming [4]. All of these algorithms have been successfully applied to the variety of problems in the field of combinatorial optimization [5]. Evolutionary algorithms are based on the Darwinian theory of evolution.

The population of individuals which represent possible solutions to an optimization problem evolve toward a better solution. To measure the quality of a solution, the fitness function is defined, which is always problem dependent. The evolution usually starts from a pool of randomly selected individuals and then, by utilizing evolutionary operators, new and better population is generated. Main evolutionary operators are selection and recombination. Selection is a process of selecting individuals that will produce a new generation. Recombination consists of two operators: crossover and mutation. Crossover combines two or more parent individuals to form one or more child individuals, that inherit parents' good characteristics. Mutation is a random change of individual alleles in an individual [6].

Main difference between all evolutionary algorithms is the way of individual encoding and evolutionary operators implementation. The genotype in genetic algorithm is usually represented as a string of bits, array of floating point numbers or a permutation vector. In genetic programming, the genotype is encoded as a tree structure. That tree structure is built by functional nodes and terminal nodes. The result of genetic programming is called a program. One of possible problems in genetic programming is bloat and there are numerous methods how to avoid it. After some generations, the search for better programs halts as the programs become too large. Bloat can be caused by inefficient code, e.g. two times $x = x + 1$, instead of $x = x + 2$, or a large calculation that is multiplied by zero in the end.

Cartesian genetic programming [7] is a type of genetic programming where a genotype is encoded as a graph structure. It is called Cartesian because functional nodes are organized in rows and columns, which reminds to a Cartesian system. This type of genetic programming solves bloat problem as well as the number of individuals in population. The genotype length is constant size and because of that the bloat problem is inherently solved. Typically, evolution strategy ES-(1+4) is used in Cartesian genetic programming. For more details on evolutionary algorithms, we refer readers to [6].

B. Swarm Algorithms

Swarm algorithms consist of ant algorithms [8], particle swarm algorithm [9], bee algorithms [10], and others. Some swarm algorithms are inspired by experiments made by biologists or by software imitation of swarms in computer graphics. Common to the all algorithms is a swarm population consisting of individuals of different characteristics. For more details on ant algorithms refer to [11] and bee algorithms refer to [12]. In the next paragraphs the particle swarm algorithm will be described.

Particle swarm optimization algorithm is created to simulate a flock of flying birds. Flock of birds is represented as a set of particles and each particle must satisfy several rules according to the neighbor particles: avoiding collision, speed harmonization, and low distance. Those rules determine social aspect of particles. Each particle is described by velocity and position. Position is a possible solution of a problem, a velocity vector changes the particle position according to the previous rules. For more information on particle swarm optimization algorithm refer to [13].

C. Other Algorithms

Other algorithms relevant for this paper are artificial immune system algorithms and cellular evolutionary algorithm. Each of these algorithms are specific and have different individual encoding or solution finding method.

Artificial immune system algorithms is a family of algorithms inspired by human immune system. Clonal Selection Algorithm (CLONALG) is the most used algorithm from this family. The algorithm begins by creating initial population of antibodies. The antibodies are made by some random mechanism. Then, affinity of each antibody is measured. By selection operator, according to affinity, antibodies are selected for cloning. The operator of hyper-mutation is applied to clones in order to improve fitness between antibody and antigen. This procedure is repeated until termination condition is reached. To read more about CLONALG, we refer interested readers to [14].

A cellular evolutionary algorithm is a type of evolutionary algorithm in which individuals cannot mate arbitrarily, but every one interacts with its close neighbors on which a basic evolutionary algorithm is applied. A cellular evolutionary algorithm usually evolves a structured bi-dimensional grid of individuals, although other topologies are also possible. In this grid, cluster of similar individuals are naturally created during evolution, promoting exploration in their boundaries, while exploitation is mainly performed by direct competition and merging inside them. This reproductive cycle is executed inside the neighborhood of each individual and consists in selecting two parents among its neighbors according to a certain criterion, applying the variation of evolutionary operators. For further information about cellular evolutionary algorithm, we refer readers to [15].

III. Modern Cryptography

Modern cryptography designs cryptographic algorithms that are assumed to be hard to break by an adversary. It is divided into symmetric key cryptography, asymmetric key cryptography, and hash functions [16]. Pseudo-random number generators can be singled out as separate algorithm that have relevance to this paper.

A. Asymmetric Key Cryptography

This type cryptography is also known as public key cryptography. In public key cryptography two keys are used, one for encryption and other for decryption. The encryption key is a public key and the decryption key is private. Public key cryptography can be divided into three groups: cryptosystems based on factorization problem, cryptosystems based on discrete logarithm problem and others.

The most famous public key cryptosystem is RSA, called after its creators [17].The security of the RSA cryptosystem is based on the difficulty of the integer factorization problem. In the process of encryption, modular exponentiation is used. Let p and q be prime numbers. Then $n = pq$, where $p, q \in \mathcal{Z}$, $de \equiv 1 \mod \varphi(n)$ and $\varphi(n)$ is Euler function. Public key is then defined as (n, e) and private key as (p, q, d). The RSA algorithm is believed to be secured as long as large p and q are used.

Famous cryptosystems or protocols based on discrete logarithm problem are Diffie-Hellman [18] or ElGamal [19]. Another large family cryptosystems are elliptic curve discrete logarithm problems (ECDLP). One representative of that family is Menzes-Vanstone cryptsystem [20]. Other public key cryptosystems are Merkle-Hellman, McEliece and NTRU.

B. Symmetric Key Cryptography

Symmetric key cryptography uses the same key for encryption and decryption, therefore some mechanism to keep those keys secret is needed. This type of cryptography is called symmetric because communication participants have the same secret key. Symmetric key cryptography's main advantage over asymmetric key cryptography is the fact that it is much faster. Symmetric key cryptography can be divided into block and stream ciphers.

Block ciphers take as an input block of plaintext and a key, and the output is a block of the ciphertext of the same size as the plaintext. Stream ciphers create an arbitrary long stream of key material, which is combined with the plaintext bit by bit. In a stream cipher, the output is created based on a hidden internal state which changes as the cipher changes.

The design of block ciphers relies on two fundamental principles: confusion and diffusion. Confusion means that each

digit of ciphertext should depend on several parts of the key. Diffusion means that if a single bit of a plaintext is changed, then, statistically, half of the bits in the ciphertext should change and vice versa.

An S-box or a substitution box is a basic component of symmetric key algorithms. The main purpose of using an S-box is to introduce a confusion. The nonlinearity property of S-boxes is one of the most important cryptographic criteria because cryptographic algorithm should be resistant to linear cryptanalysis [21]. S-box (n, m) consists of n input variables and m output variables. These m outputs can be viewed like m Boolean functions. If nonlinearity of a Boolean function is equal to maximum then it is called bent function. The nonlinearity bound, show in 1, is called the covering radius bound and is strict for bent Boolean functions.

$$N_f \leq 2^{n-1} - 2^{\frac{n}{2}-1}. \quad (1)$$

Other criteria which is considered in Boolean or vectorial functions are algebraic degree, balancedness, resiliency, algebraic immunity, and others [21]. If all criteria are used simultaneously then the problem of finding the best Boolean function or an S-box is a multi-objective problem. To inform more about definitions or mathematics background see [22], [23].

C. Pseudo-random Number Generator

A pseudorandom number generator (PRNG) is a deterministic algorithm for generating a sequence of numbers that have the properties of random numbers. The sequence obtained by the pseudorandom number generator is not truly random because it is determined by small set of initial values, called the state. Pseudorandom number generators play important role in the area of cryptography. General security requirements of PRNGs are listed in below [24]:

- the random numbers should not show any statistical weaknesses,
- the knowledge of subsequences of random numbers shall not allow to compute predecessors or successors practically or to guess them with non-negligibly larger probability than without knowledge of these subsequences,
- it shall not be practically feasible to compute preceding random numbers from the internal state or to guess them with non-negligibly larger probability than without knowledge of the internal state and
- it shall not be practically feasible to compute future random numbers from the internal state or to guess them with non-negligibly larger probability than without knowledge of the internal state.

The most famous and the simplest PRNG is the linear congruential generator (LCPRNG) given in equation 2. The s_0 is called seed. Another PRNGs are RSA PRNG and Blum-Blum-Shub PRNG.

$$s_{i+1} = (as_i + b) \ mod \ M, a, b \in \{1, ..., M-1\}. \quad (2)$$

IV. Evolutionary Computation in Cryptography

In next subsections the state-of-the-art overview of the evolutionary computation algorithms will be given.

A. Evolutionary Computation in Asymmetric Key Cryptography

Two problems in asymmetric key cryptography chosen to be described are finding short addition chains and cracking of the Merkle-Hellman cryptosystem using genetic algorithm.

1) Addition Chain: Field of the modular exponentiation has several important applications in cryptography. Well-known public key cryptosystem such as RSA adopt modular exponentiation. Modular exponentiation is defined as follows:

$$B = A^c \ mod \ p. \quad (3)$$

where A is an integer in the range $[1, ..., p-1]$ and c and p are an arbitrary positive integers. In cryptosystems, p is a large positive prime number defined in III-A. One possible way of reducing the computational load of Eq. (3) is to minimize the total number of multiplications required to compute the exponentiation. Since the exponent in Eq. (3) is additive, the problem of computing powers of the base A can be formulated as an addition calculation, for which so-called addition chains are used. For example, if A^{50} wants to be calculated, the traditional procedure would require 50 multiplications. If the addition chain is used, then only 7 multiplications are required: $[1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \rightarrow 32 \rightarrow 48 \rightarrow 50]$.

Usual deterministic algorithm is the binary method. The exponent is written in binary representation (of length d) and then each occurrence of digit 1 is replaced with the letters "SM" and each digit 0 with letter "S", where "S" stands for squaring and "M" stands for multiplication. After all digits are replaced, the first "SM" occurrence on the left is crossed. This method is deterministic but it does not give the optimal solution. There are two version of binary method: left-to-right and right-to-left. Both methods have same number of operations: d times squaring and multiplications as much as there is number of digit 1.

In [25] experiment with the Particle Swarm Optimization algorithm in order to find optimal short addition chains. Authors in [26] implement the Ant Colony Optimization algorithm on a System on s Chip (SoC) in order to speed up the modular exponentiation in cryptographic applications. In [27] use a GA to find minimal Brauer chains where a Brauer chain is an addition chain in which each member uses the previous member as a summand. In [28] authors investigate the usage of evolutionary programming for minimizing the length of addition chains.

In [29] authors proposed a genetic algorithm to find short addition chains for a given exponent. The main paper contributions were a new representation of solutions, design several mutation and crossover operators designed to improve convergence and design a special part of algorithm called *repair heuristics* that is believed to be an integral part of the algorithm.

Internally the alleles were pair of positions (i_1, i_2) that hold values (n_1, n_2) which form value on given position.

That type of encoding is called encoding with *summand positions*. *Repair heuristics* is strategy needed to recreate invalid individual. The genetic algorithm in that paper is described as follows. First, set zeroth element to 1 and the first element to 2. Then uniformly at random select between all minimal subchains consisting of three elements and a random choice of the second element. With a probability equal to $3/5$ double the elements until they reach half of the exponent size. Check weather the current element and any previous element sum up to the exponent value. Uniformly at random choose mechanisms given in paper to obtain the next value in the chain, under the constraint that it needs to be smaller than the exponent value. The used fitness function was the number of elements in the chain.

Authors showed that with proposed genetic algorithm is possible to find better solution than with the binary method.

2) Merkle-Hellman Cryptosystem: Merkle-Hellman cryptosystem is a public key cryptosystem based on a concrete case of the knapsack problem [30]. The knapsack problem is defined as follows: for a given set $\{v_1, v_2, ..., v_n\}$ of n integers and an integer V, find an array $m = (\epsilon_1, \epsilon_2, ..., \epsilon_n)$, where $\epsilon_i \in \{0, 1\}$, so that $\epsilon_1 v_1 + ... + \epsilon_n v_n = V$, if such m exists. This problem is NP-complete, which means that there is no polynomial algorithm to solve that problem. However, the special case of the knapsack problem is the super-increasing sequence, which can be solved in polynomial time. Super-increasing sequence is satisfied if $v_j > v_1 + v_2 + ...v_{j-1}$ for $j = 2, 3, ..., n$. One example of the super-increasing sequence is $v_i = 2^{i-1}$. The main idea of the Merkle-Hellman cryptosystem is to mask the super-increasing sequence to look like a random sequence. The message receiver should know how to remove the mask, and than solve the super-increasing problem. Masking is done by modular multiplication.

Shamir [31] showed existence of polynomial algorithm for breaking Merkle-Hellman cryptosystem. However, there is a genetic algorithm twice faster. In [32] authors proposed a genetic algorithm as a method to crack the cryptosystem. The authors proposed a new selection and crossover method.

The selection function chooses the selection candidates, which the most satisfy the fitness function, i.e., their fitness values are higher than those of others. In case the population size is L only $L/5$ solution candidates is chosen. Exactly these solution candidates form new generations.

The crossover function receives the population of the solution candidates. From this population solution candidates with $t1$ and $t2$ numbers are chosen in pairs by means of a random generator taking into account that $t1$ and $t2$ do not coincide with each other and the used pair is not repeated. Each solution candidate is divided in two parts (at the mid point). To explore more about given algorithm refer to [32].

B. Evolutionary Computation in Symmetric Key Cryptography

In next subsections the state-of-the-art overview of the evolutionary computation algorithms applied to Boolean and vectorial functions construction will be shown.

1) Cryptographic Boolean Functions: Boolean functions represent an important primitive in the design of various cryptographic algorithms. Three main approaches for generating Boolean functions for cryptographic usages are distinguished: algebraic constructions, random generation, and heuristic constructions.

There exists a number of works that examine Boolean functions in cryptography and their generation with evolutionary computation techniques. Here will be enumerated only relevant papers.

[33] compare the effectiveness of Cartesian genetic programming and genetic programming approach when looking for highly nonlinear balanced Boolean functions of eight inputs. Boolean function is balanced if there is same number of ones and zeros. Nonlinearity is minimal Hamming distance between Boolean function and its all affine functions. [34] investigate several evolutionary algorithms in order to evolve Boolean functions with different values of the correlation immunity property. In the same paper, the authors also discuss the problem of finding correlation immune functions with minimal Hamming weight, but they experiment with only one size of Boolean functions. More extensive investigation on finding correlation immune Boolean functions with minimal Hamming weight and different sizes is conducted by [35].

[36] use Particle Swarm Optimization to find Boolean functions with good trade-offs of cryptographic properties for dimensions up to 12. The same authors use genetic algorithms where the genotype consists of the Walsh-Hadamard values in order to evolve semibent (plateaued) Boolean functions [37]. Semibent Boolean function has all values in Walsh-Hadamard spectrum in $\{0, 2^{\frac{n+2}{2}}, -2^{\frac{n+2}{2}}\}$, where n is input space dimensionality.

[38] conduct a detailed analysis of the efficiency of a number of evolutionary algorithms and fitness functions for Boolean functions with 8 inputs.

In [39] authors used several evolutionary algorithms to evolve Boolean functions with several cryptographic criteria. Authors used genetic algorithm, evolutionary strategy, genetic programming and Cartesian genetic programming. Individuals were, according to the algorithm, encoded by string of bits whose values define a truth table of th Boolean function, trees of Boolean primitives which are then evaluated according to the truth table they produce or directed graphs that are also evaluated in accordance to the truth table they produce.

Authors used two objective functions. First function was the nonlinearity where the goal is maximization. In the work, truth table was transformed by using Walsh-Hadamard transformation as follows:

$$W_f(\vec{a}) = \sum_{\vec{x} \in F_2^n} (-1)^{f(\vec{x}) \oplus \vec{a} \cdot \vec{x}} \tag{4}$$

where $\vec{a} \cdot \vec{x}$ is the inner product of two vectors. With that objective function the goal was to find the bent function. Although bent functions are not appropriate as parts of filter and combiner generators, there is nevertheless motivation in their generation. Filter and combiner generators are generators of random sequence which uses linear feedback shift registers (LFSR). The first reason is that this problem can be used as a benchmark to test various evolutionary algorithms. The second reason lies in the fact that using bent functions it is possible

to build highly nonlinear functions that are balanced [22]. The second objective was combination of balancedness and nonlinearity. The goal was to find highly nonlinear function that is balanced.

In that paper authors showed that genetic programming and Cartesian genetic programming performed the best, which indicates that the truth table representation in not the most appropriate one when evolving cryptographically suitable Boolean functions.

In [40] authors investigate the efficiency of two immunological algorithms: CLONALG and optimization immune algorithm with elitism. For all algorithms is experimented with two different representations for encoding a Boolean function: string of bits and floating point representation. Same objective functions were used as in [40].

By using those algorithms and individual encodings, it is impossible to conclude whether the string of bits or floating point encoding should be used. Furthermore, the results show that the optimization immune algorithm performs better than the CLOANLG, but both perform slightly worse than the genetic algorithm and evolutionary strategy.

2) Evolution of S-boxes: Substitution boxes (S-boxes) play an important role in many modern-day cryptographic algorithms, more commonly known as ciphers. Without carefully chosen S-boxes, such ciphers would be easier to break. In the process of the design of S-boxes (similarly as in the design of Boolean functions), one can roughly follow three directions, namely, algebraic constructions, random search, and heuristics.

There are several unique S-box representations. An (n, m)-function *F* can be represented as a list of values (lookup table - LUT), with each value ranging from 0 to $2^m - 1$. Alternatively said, an (n, m)-function can be implemented as a lookup table with 2^n words of m bits each. When $n = m$ it is usual that the S-box is bijective, i.e., that each value in the output appears exactly once.

A Boolean function f on $\mathbb{F}_2^n$ is represented by a truth table (TT), which is a vector $(f(\vec{0}), \ldots, f(\vec{1}))$ that contains the function values of f, ordered lexicographically, i.e., $\vec{a} \leq \vec{b}$, where $\vec{a}$ and $\vec{b}$ are two input entries for the truth table [22]. An S-box can be represented in the truth table form as a matrix of dimension $2^n \times m$ where each column m represents one Boolean function (i.e., one coordinate function).

The Walsh-Hadamard transform of an (n, m)-function *F* equals [23]:

$$W_F(\vec{a}, \vec{v}) = \sum_{\vec{x} \in \mathbb{F}_2^n} (-1)^{\vec{v} \cdot F(\vec{x}) \oplus \vec{a} \cdot \vec{x}}, \tag{5}$$

where $\vec{a} \in \mathbb{F}_2^n$ and $\vec{v} \in \mathbb{F}_2^{m*}$.

An (n, m)-function F is called *balanced* if it takes every value of $\mathbb{F}_2^m$ the same number 2^{n-m} of times. Balanced (n, n)-functions are permutations on $\mathbb{F}_2^n$ [23].

[41] experiment with a GA that works in a reverse way in order to generate bijective S-boxes of dimensions from 8×8 to 16×16. They seed the initial S-boxes population with solutions based on the finite field inversion method and then evolve them to find new solutions. The same authors use a modified immune algorithm to generate 8×8 S-boxes that are balanced, with high nonlinearity, and low δ-uniformity [42].

[43] explore how to generate S-boxes of size 8×8 with a better resistance against side-channel attacks (SCA) as measured with the transparency order property. Next, [44], [45] investigate side-channel resilience of 4×4 S-boxes as well as when considering the confusion coefficient property. [46] use genetic algorithms to evolve S-boxes with better SCA resilience and they implement such S-boxes in both software and hardware settings in order to properly evaluate them. Finally, [47] experiment with the modified transparency order property in order to achieve S-boxes with better SCA resistivity.

[48] use Cartesian genetic programming (CGP) and genetic programming (GP) to evolve S-boxes where they discuss how to obtain permutation-based encoding with those algorithms.

In [49] authors experimented with three different S-box representations: truth table, Walsh-Hadamard transform coefficients and the lookup table representation where in the case that $n = m$ an S-box can be represented as a permutation.

The goal was to evolve highly nonlinear S-box with different dimensions. The authors proposed a new cost function that is faster than those in given related work. This cost function is defined as a two-component vector with the nonlinearity as the first component and a cost associated with a part of the Walsh-Hadamard spectrum as the second component; when using in a single criterion optimizer, to determine better solutions, a hierarchical comparison should be performed.

C. Evolving Cryptographic Pseudorandom Number Generators

Random number generators are used in a range of applications spanning from producing simple values and adding randomness to programs, over online betting to various cryptographic applications. One real-world application of pseudorandom number generators in cryptography is to use them for masking as a countermeasure against side channel attacks [50]. When used for such a purpose, those generators needs to be extremely fast and small when implemented in hardware. To obtain a generator with such characteristics, "expensive" operations like multiplication or addition should be avoided as much as possible.

John Koza used genetic programming (GP) to evolve programs that output random numbers [51]. As a fitness function he used the notion of information entropy as defined by Shannon and the end result was a program that was able to accept a sequence of consecutive integer values and transform it into random binary digits.

Hernandez, Seznec, and Isasi used GP to evolve random number generators where they used the strict avalanche criterion (SAC) as a fitness function [52]. Martinez et al. designed a pseudorandom number generator suitable for cryptographic usage by means of GP. The obtained generator, Lamar was tested with a number of tests where the input values were obtained via a counter function.

In [50] authors used Cartesian genetic programming to evolve a pseudorandom number generator. Functional nodes consisted of logical functions like AND, NOT, XOR, operations RR/RL (rotate right or rotate left) and SR/SL (shift right

or shift left), and a function P which is shuffle function. The test for randomness was the bias of the output sequence, which is defined as a ratio of zeros and ones in the sequence.

As the fitness function the entropy is used, where the goal is to maximize entropy in order to achieve randomness. With that fitness function and by using Cartesian genetic programming authors evolved pseudorandom number generators that are extremely small and fast in hardware implementation because they don't rely on expensive operations like multiplication or addition.

V. Conclusion

In this paper, we showed how evolutionary computation can be successfully used in cryptography. We present different approaches in asymmetric key cryptography, symmetric key cryptography and pseudo-random number generator. In asymmetric key cryptography, evolutionary computation can speed-up exponentiation calculation or be used for breaking Merkle-Hellman cryptosystem. In symmetric key cryptography, evolutionary computation can build Boolean functions or S-boxes which satisfy various characteristics simultaneously. Finally, evolutionary computation can be successfully used in evolving pseudorandom number generator.

Acknowledgments

This work has been supported in part by Croatian Science Foundation under the project IP-2014-09-4882.

References

[1] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[2] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[3] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.

[4] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence*. John Wiley & Sons, 2006, vol. 1.

[5] M. Affenzeller, S. Wagner, S. Winkler, and A. Beham, *Genetic algorithms and genetic programming: modern concepts and practical applications*. Crc Press, 2009.

[6] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin Heidelberg New York, USA, 2003.

[7] J. F. Miller, "Cartesian genetic programming," in *Cartesian Genetic Programming*. Springer, 2011, pp. 17–34.

[8] O. Cordón García, F. Herrera Triguero, and T. Stützle, "A review on the ant colony optimization metaheuristic: Basis, models and new trends," *Mathware & soft computing. 2002 Vol. 9 Núm. 2 [-3]*, 2002.

[9] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. part i: background and development," *Natural Computing*, vol. 6, no. 4, pp. 467–484, 2007.

[10] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 8, no. 1, pp. 687–697, 2008.

[11] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[12] X.-S. Yang, "Engineering optimizations via nature-inspired virtual bee algorithms," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer, 2005, pp. 317–323.

[13] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.

[14] D. DasGupta, "An overview of artificial immune systems and their applications," in *Artificial immune systems and their applications*. Springer, 1993, pp. 3–21.

[15] M. Tomassini, "Cellular evolutionary algorithms," in *Simulating Complex Systems by Cellular Automata*. Springer, 2010, pp. 167–191.

[16] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.

[17] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[18] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[19] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[20] N. Koblitz, A. Menezes, and S. Vanstone, "The state of elliptic curve cryptography," in *Towards a quarter-century of public key cryptography*. Springer, 2000, pp. 103–123.

[21] C. Carlet, "Boolean functions for cryptography and error correcting codes," *Boolean models and methods in mathematics, computer science, and engineering*, vol. 2, pp. 257–397, 2010.

[22] ——, "Boolean Functions for Cryptography and Error Correcting Codes," in *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, 1st ed., Y. Crama and P. L. Hammer, Eds. New York, NY, USA: Cambridge University Press, 2010, pp. 257–397.

[23] ——, "Vectorial Boolean Functions for Cryptography," in *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, 1st ed., Y. Crama and P. L. Hammer, Eds. New York, NY, USA: Cambridge University Press, 2010, pp. 398–469.

[24] F. Özkaynak, "Cryptographically secure random number generator with chaotic additional input," *Nonlinear Dynamics*, vol. 78, no. 3, pp. 2015–2020, 2014.

[25] A. León-Javier, N. Cruz-Cortés, M. A. Moreno-Armendáriz, and S. Orantes-Jiménez, "Finding minimal addition chains with a particle swarm optimization algorithm," in *Mexican International Conference on Artificial Intelligence*. Springer, 2009, pp. 680–691.

[26] N. Nedjah and L. de Macedo Mourelle, "High-performance soc-based implementation of modular exponentiation using evolutionary addition chains for efficient cryptography," *Applied Soft Computing*, vol. 11, no. 7, pp. 4302–4311, 2011.

[27] A. Rodriguez-Cristerna and J. Torres-Jimenez, "A genetic algorithm for the problem of minimal brauer chains," in *Recent Advances on Hybrid Intelligent Systems*. Springer, 2013, pp. 481–500.

[28] S. Domínguez-Isidro, E. Mezura-Montes, and L.-G. Osorio-Hernández, "Evolutionary programming for the length minimization of addition chains," *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 125–134, 2015.

[29] S. Picek, C. A. C. Coello, D. Jakobovic, and N. Mentens, "Evolutionary algorithms for finding short addition chains: Going the distance," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2016, pp. 121–137.

[30] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[31] A. Shamir, "A polynomial-time algorithm for breaking the basic merkle-hellman cryptosystem," *IEEE transactions on information theory*, vol. 30, no. 5, pp. 699–704, 1984.

[32] Z. Kochladze and L. Beselia, "Cracking of the merkle–hellman cryptosystem using genetic algorithm," 2016.

[33] S. Picek, D. Jakobovic, J. F. Miller, E. Marchiori, and L. Batina, "Evolutionary Methods for the Construction of Cryptographic Boolean Functions," in *Genetic Programming - 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, 2015, pp. 192–204.

[34] S. Picek, C. Carlet, D. Jakobovic, J. F. Miller, and L. Batina, "Correlation Immunity of Boolean Functions: An Evolutionary Algorithms Perspective," in *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*, 2015, pp. 1095–1102.

[35] S. Picek, S. Guilley, C. Carlet, D. Jakobovic, and J. F. Miller, "Evolutionary Approach for Finding Correlation Immune Boolean Functions of Order t with Minimal Hamming Weight," in *Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings*, 2015, pp. 71–82.

[36] L. Mariot and A. Leporati, "Heuristic Search by Particle Swarm Optimization of Boolean Functions for Cryptographic Applications," in *Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings*, 2015, pp. 1425–1426.

[37] ——, "A Genetic Algorithm for Evolving Plateaued Cryptographic Boolean Functions," in *Theory and Practice of Natural Computing - Fourth International Conference, TPNC 2015, Mieres, Spain, December 15-16, 2015. Proceedings*, 2015, pp. 33–45.

[38] S. Picek, D. Jakobovic, J. F. Miller, L. Batina, and M. Cupic, "Cryptographic Boolean functions: One output, many design criteria," *Applied Soft Computing*, vol. 40, pp. 635–653, 2016.

[39] ——, "Cryptographic boolean functions: One output, many design criteria," *Applied Soft Computing*, vol. 40, pp. 635–653, 2016.

[40] S. Picek, D. Sisejkovic, and D. Jakobovic, "Immunological algorithms paradigm for construction of boolean functions with good cryptographic properties," *Engineering Applications of Artificial Intelligence*, 2016.

[41] G. Ivanov, N. Nikolov, and S. Nikova, "Reversed genetic algorithms for generation of bijective s-boxes with good cryptographic properties," *Cryptography and Communications*, vol. 8, no. 2, pp. 247–276, 2016.

[42] ——, "Cryptographically Strong S-Boxes Generated by Modified Immune Algorithm," in *Cryptography and Information Security in the Balkans - Second International Conference, BalkanCryptSec 2015, Koper, Slovenia, September 3-4, 2015, Revised Selected Papers*, 2015, pp. 31–42.

[43] S. Picek, B. Ege, L. Batina, D. Jakobovic, L. Chmielewski, and M. Golub, "On Using Genetic Algorithms for Intrinsic Side-channel Resistance: The Case of AES S-box," in *Proceedings of the First Workshop on Cryptography and Security in Computing Systems*, ser. CS2 '14. New York, NY, USA: ACM, 2014, pp. 13–18. [Online]. Available: http://doi.acm.org/10.1145/2556315.2556319

[44] S. Picek, B. Ege, K. Papagiannopoulos, L. Batina, and D. Jakobovic, "Optimality and beyond: The case of 4x4 S-boxes," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014, Arlington, VA, USA, May 6-7, 2014*, 2014, pp. 80–83. [Online]. Available: http://dx.doi.org/10.1109/HST.2014.6855573

[45] S. Picek, K. Papagiannopoulos, B. Ege, L. Batina, and D. Jakobovic, "Confused by Confusion: Systematic Evaluation of DPA Resistance of Various S-boxes," in *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, 2014, pp. 374–390.

[46] B. Ege, K. Papagiannopoulos, L. Batina, and S. Picek, "Improving DPA resistance of S-boxes: How far can we go?" in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 2013–2016.

[47] S. Picek, B. Mazumdar, D. Mukhopadhyay, and L. Batina, "Modified Transparency Order Property: Solution or Just Another Attempt," in *Security, Privacy, and Applied Cryptography Engineering - 5th International Conference, SPACE 2015, Jaipur, India, October 3-7, 2015, Proceedings*, 2015, pp. 210–227.

[48] S. Picek, J. F. Miller, D. Jakobovic, and L. Batina, "Cartesian Genetic Programming Approach for Generating Substitution Boxes of Different Sizes," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO Companion '15. New York, NY, USA: ACM, 2015, pp. 1457–1458. [Online]. Available: http://doi.acm.org/10.1145/2739482.2764698

[49] S. Picek, M. Cupic, and L. Rotim, "A new cost function for evolution of s-boxes," *Evolutionary Computation*, vol. 24, no. 4, pp. 695–718, 2016.

[50] S. Picek, D. Sisejkovic, V. Rozic, B. Yang, D. Jakobovic, and N. Mentens, "Evolving cryptographic pseudorandom number generators," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 613–622.

[51] J. R. Koza, "Evolving a computer program to generate random numbers using the genetic programming paradigm." in *ICGA*. Citeseer, 1991, pp. 37–44.

[52] J. C. Hernandez, A. Seznec, and P. Isasi, "On the design of state-of-the-art pseudorandom number generators by means of genetic programming," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2. IEEE, 2004, pp. 1510–1516.