On Evolutionary Computation Methods in Cryptography

S. Picek*, M. Golub **
Faculty of Electrical Engineering and Computing, Zagreb, Croatia
*stjepan@computer.org, **marin.golub@fer.hr

Abstract - Evolutionary computation algorithms represent a range of problem-solving techniques based on principles of biological evolution, like natural selection and genetic inheritance. Such algorithms can be used to solve a variety of difficult problems, among which are those from the area of cryptography. Examples of such an approach include the evolving hash functions or creation of a new block cipher. First results in this area have emerged over 30 years ago, and in recent years there has been an increased interest in this area. Still, some problems like problem formulation and representation remain open. The purpose of this paper is to give a survey of cryptographic applications that can be developed with the help of evolutionary computation methods, and to address their applicability to the real-world scenarios.

I. INTRODUCTION

Computational intelligence represents a set of nature-inspired computational methodologies and approaches to address complex problems of the real world applications to which traditional methodologies and approaches are ineffective. These methods have found a variety of applications in the field of optimization problems. Some of those applications belong to the area of information security. First research papers dealing with that subject appeared in 1979 and presented cryptanalysis of simple substitution cipher by means of relaxation algorithms [8]. Since then, as a problem solving techniques in information security, several areas of biologically inspired computation methods that belong to computational intelligence gained attention. Although in this paper we present only evolutionary computation methods, it is justified to mention several others, like artificial neural networks, DNA computing, and cellular automata as possible alternatives. In this paper, we limit our attention only to applications of evolutionary computation methods to modern cryptography. For more details on evolutionary computation methods in cryptanalysis refer to [13], and for information about evolutionary computation methods applied to classical cryptology refer to [2].

This paper is not intended as a detailed survey of possible applications of evolutionary computation in cryptography. Rather, it aims to give an introduction to the possibilities of evolutionary computation methods when applied to cryptography.

We begin this paper by discussing relevant theory in Section 2. Section 3 describes the applications of the evolutionary computation methods used in cryptography, and finally, Section 4 draws a conclusion.

II. PRELIMINARY

A. Evolutionary Computation Methods

Evolutionary computation uses iterative progress, such as growth or development in a population. This population is then selected in a guided random search to achieve the desired goal. Such processes are often inspired by biological mechanisms of evolution.

In following paragraphs, we briefly explain genetic algorithms (GAs), genetic programming (GP), tabu search (TS), and simulated annealing (SA).

1) Genetic Algorithms

Genetic algorithm is a search heuristics that mimics the process of natural evolution. Genetic algorithms are based on the Darwinian theory of evolution. Genetic algorithms have been invented by J. Holland in 1960s, and since then they have been successfully applied to the variety of problems in the field of combinatorial optimization. Because of their popularity, numerous variations of genetic algorithms arose since their invention.

In genetic algorithms, 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. A fitness function is always problem dependant. The evolution in genetic algorithm usually starts from a pool of randomly selected individuals and then, by utilizing the GA operators, a new and better population is generated.

Main genetic algorithm operators are selection, crossover, and mutation. Selection is a process of selecting individuals that will produce a new generation. Crossover works by combining two or more parent solutions to form one or more child solutions that have their good characteristics. Mutation is a random change of individual alleles in an individual [11] [18].

2) Genetic Programming

Genetic programming represents evolutionary methodology inspired by biological evolution to find computer programs. They are used to optimize a population of computer programs according to a fitness landscape. Genetic programming started to develop in 1950s, but it expanded in 1990s when J. Koza pioneered

the application of genetic programming in various search problems. Genetic programming has also been applied to evolvable hardware.

Computer programs that are developed by GP are traditionally represented as a tree structure. Every tree node has an operator function and every terminal node has an operand. Main operators used in genetic programming are crossover and mutation. Crossover is done by switching one of the nodes of an individual with one node from another individual. Mutation is used to replace a whole node or node's information of an individual [10] [18].

3) Tabu Search

Tabu search is an optimization method that belongs to the class of local search techniques. Tabu search works in a way that enhances the performance of a local search method. Tabu search uses memory structures where it stores the solutions. The most important of memory structures is a tabu list. Once the potential solution is determined, it is put on a tabu list so the algorithm cannot visit that possibility repeatedly.

A tabu search uses a local search procedure to iteratively move from one solution to another which is in the proximity of the first solution. To explore regions of the search space that would be left unexplored, tabu search modifies the local structure of each solution as the search progresses.

Tabu search was invented by W. Glover [18].

4) Simulated Annealing

Simulated annealing is a generic probabilistic metaheuristics inspired by the cooling processes of molten metal. It was invented by S. Kirkpatrick as an adaptation of the Metropolis-Hastings algorithm invented in 1950s. Simulated annealing combines hill-climbing technique with the probabilistic acceptance of non improving moves.

In simulated annealing, each point of search space is analogous to a state of a physical system, and the function that needs to be minimized is analogous to the internal energy of the system in that state. The goal is to bring the system from arbitrary state to the state with minimal energy.

The search starts at some initial state with a control parameter known as the temperature. The search tries to avoid local minima by jumping out of them early in the computation. Toward the end of the computation, when the temperature, or probability of accepting a worse solution, is nearly zero, the search simply seeks the bottom of the local minimum. The chance of getting a good solution can be traded off with computation time by slowing down the cooling schedule. The slower the cooling, the higher is the chance of finding the optimum solution, but the run time is also longer [18].

B. Modern Cryptology

Modern cryptography designs cryptographic algorithms that are assumed to be hard to break by an adversary.

Modern cryptography can be divided into several areas. Some of those areas that are relevant for this paper are discussed in the following paragraphs [9] [14].

Modern cryptography can be informally divided into symmetric key cryptography, public key cryptography, and hash functions. Additionally, cryptographic primitives and pseudorandom number generators can be singled out as separate areas that have relevance to this paper.

1) Symmetric Key Cryptography

Symmetric-key cryptography refers to the encryption methods in which both the sender and receiver share the same key. In these algorithms encryption key can be calculated from decryption key and vice versa. In these algorithms as long as the communication needs to remain secret, the key must remain secret. Symmetric key cryptography's main advantage over public key cryptography is the fact that it is much faster. Symmetric key cryptography can be generally divided into block ciphers and stream ciphers [14].

a) Block Ciphers

Block ciphers take as an input a block of plaintext and a key, and the output is a block of the ciphertext of the same size.

b) Stream Ciphers

Those 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.

c) Hash Functions

Hash functions maps an arbitrary length input into a fixed length output. Hash function to be cryptographically secure needs to fulfill a number of prerequisites.

2) Public Key Cryptography

In these systems, the key used for encryption is different from the key used for decryption. These algorithms are called public key because the encryption key can be made public. The system remains safe as long as the private key remains secret. Public key cryptography is most often based on the computational complexity of the hard mathematical problems from the number theory.

a) RSA Cryptosystem

RSA is the first algorithm known to be suitable for encryption, and digital signing. The security of the RSA cryptosystems is based on the difficulty of the integer factorization problem. In the process of encryption and decryption, modular exponentiation is used, and factorization problem is utilized to create the trapdoor function. It is believed to be secure if sufficiently long keys are used.

3) Cryptographic Primitives

Primitives are algorithms that have basic cryptographic properties. Primitives provide fundamental properties that are used to develop more complex cryptographic protocols. Cryptographic primitives, on their own, are quite limited, but when combined in security protocols, then more than one security requirement can be addressed.

4) Pseudorandom number generator

A pseudorandom number generator (PRNG) is an 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. It is necessary to conduct the mathematical analysis of a sequence before it can be determined that is sufficiently random for the intended use.

III. EVOLUTIONARY COMPUTATION IN CRYPTOGRAPHY

A. *ICIGA system*

ICIGA (Improved Cryptography Inspired by Genetic Algorithms) system represents an improvement of a system "Genetic Algorithms Inspired Cryptography" done by the same authors [15]. ICIGA is a block cipher system where secret key is generated during each session via random process. The length of the key and the block size are the parameters adjustable by the user of the system. Based on the key length, plaintext is divided into the parts of the same size. The first part is used to generate secret key which will be used in ciphering of the message. The difference between the two versions of the system is in adding more transformation operations in the ICIGA system.

The algorithm for encryption can be described informally in the following way. First, break the binary encoded plaintext into parts of equal size based on the length of the key, and then break those parts in the blocks of the same size. Next, crossover and mutation operations based on the genetic algorithms are applied. Those operations work on randomly selected blocks and positions in those blocks. Based on the trace of those operations a secret key is generated. Then mask the position of the genetic algorithm operators by applying left shift. Third, apply left shift on whole part to mask the distribution of the blocks. Finally, apply same steps to the rest of the plaintext but use secret key to choose genetic algorithm operations and positions instead.

Decryption is composed of the right shift operation and the same genetic algorithm operators since the inverse of left shift is right shift, and the crossover and mutation are involutions. The operations in deciphering process should be done in reverse order.

The crossover operation is done by permuting the bits between two blocks, and the mutation is done by applying logical negation on the bits. Crossover operation can be regarded CBC (Cipher Block Chaining) mode of ciphering and mutation operation as ECB (Electronic Code Book) mode of ciphering. It is necessary to mention that no actual genetic algorithm is used, and because of that, no selection scheme or fitness function is needed. Crossover and mutation operations that draw inspiration from genetic algorithms are used for encryption and decryption.

Authors made some comparisons of the ICIGA system with other symmetric key ciphers - DES, IDEA, and AES algorithms.

Authors claim that their system is faster in encryption and decryption then AES or IDEA [16]. To our knowledge no further research of the security level of ICIGA system has been done.

B. *Evolving Hardware for RSA Systems*

Modular exponentiation represents the most important part in the RSA cryptosystems, and modular multiplication is the most time consuming operation within. When engineering hardware for those operations, it is necessary to optimize the time consumed by a single modular multiplication or reduce the total number of modular multiplication performed (or both of those). Many hardware designs using modular multiplication have a drawback called side-channel leakage. As the designs for modular exponentiation are regular and repetitive it is possible to trace the data transfers between operations, which can reveal the beginning of every long integer operation. This allows an adversary to discover the private key of the cryptosystem. Designing a hardware that fulfils a certain function consists of deriving an operational architecture within a set of constraints.

It is possible to develop new cryptographic circuits by means of evolutionary computation methods.

Authors used genetic programming and evolvable hardware to develop new cryptographic circuits in their paper [12]. Evolvable hardware refers to the hardware that can change its architecture and behavior dynamically and autonomously by interacting with its environment.

They represented individuals as register-transfer level specifications of the hardware. Each instruction in the specification is an output signal assignment. A signal is assigned the result of an expression wherein the operators are those that represent basic gates in CMOS technology of the VLSI circuit implementation and the operands are the input signals of the design. Specifications are encoded by using an array of parse trees corresponding to its signal assignments. As recombination operators, crossover and mutation are used. Fitness function adheres to the following constraints: evolved specification must obey input/output behavior, circuits must have reduced size, and the circuits must reduce signal propagation time. Authors compared the design obtained by genetic programming with some other well known, human-made designs and found that their solution is more efficient in both required hardware area and encryption/decryption throughput [12].

C. *Finding Cryptographically Sound Boolean Functions*

Significant work in this area was done by Clark in his PhD Thesis [2]. Genetic algorithms are used there to search for cryptographically sound Boolean functions.

A Boolean function describes how to determine a Boolean value output based on some logical calculation from Boolean inputs. The Boolean function is the linear combinations of S-box columns, which is an important cryptographic primitive for block and stream ciphers. Because many cryptographic properties of Boolean functions conflict with each other, the choice of functions is usually a compromise between the desired properties of the functions. High nonlinearity is one extremely

important property needed in order to reduce the effectiveness of attacks such as linear cryptanalysis.

Clark introduced technique which enables the creation of a complete list of Boolean function inputs in such way that complementing any one of the corresponding truth table positions will increase the nonlinearity of the function. Each truth table position corresponds to a unique function input. To find the list of truth table positions Clark first found the values of Walsh-Hadamard transform coefficients. First experiments were done by utilizing hill-climbing techniques on the binary truth tables.

Further, Clark experimented with genetic algorithms in attempt to find even better, more nonlinear solutions. He used genetic algorithms with binary representation where individuals are represented as binary strings. For an initial population a pool of randomly created Boolean functions was used. The fitness of the solutions is calculated based on nonlinearity of the solutions. As a recombination operator the "merge" operator is used, where it produces a single offspring from two parents based on their Hamming distance. The main idea behind that operator is to allow two good parents that have small Hamming distance to produce an offspring close to them. The mutation operator is not used because it is likely to reduce the nonlinearity of the solutions.

The results obtained in that work showed that genetic algorithms on its own, or in the combination with hill-climbing techniques, yield much better results then those obtained by a random search.

D. Evolving Block Ciphers and Cryptographic Hash Functions

Cryptographic hash functions represent important part of many security protocols. Hash functions are usually designed by experienced experts from the area of cryptography, but it is also possible to develop them from automatically obtained nonlinear functions.

It is possible to develop the compression function from block cipher by utilizing Miyaguchi-Preenel construction. For instance, that is the principle that is utilized in obtaining Whirlpool hash function.

It is possible to develop new block cipher and then on the basis of that cipher a new hash function like presented in the paper [1] [5].

To develop a block cipher, it is necessary to decide on highly-nonlinear functions that will be used in that algorithm. As criteria for estimating nonlinearity of a function avalanche effect is used. Informally, avalanche effect can be defined as the effect that minimum change of the input (one bit) changes on average, half of the output bits.

The core of their work was to design functions with nearly ideal amount of avalanche effect. As an evolutionary computation method in developing functions with that kind of properties, authors used genetic programming. Experiments were conducted after an adequate parameter set was chosen. For a function set, efficient operations that are easy to implement in software and hardware were chosen. To develop a block cipher that follows a Feistel scheme it is necessary to find a key schedule algorithm, and a round function.

After obtaining the two functions with desired properties, they used these functions as a core component of a new block cipher called Wheedham. Authors found that that cipher is secure enough if it has eight rounds, but they recommended that it has 16 rounds.

To produce a cryptographic hash function, they used a Miyaguchi-Preenel construction on a modified Wheedham cipher. Resulting hash functions was called MPW-512. Furthermore, they found that four rounds would be enough to ensure an appropriate security level of the MPW-512 hash function. Some testing in the terms of speed and security were performed and it was found that resulting hash function is competitive to best hash functions available in that time (SHA-512, and Whirlpool). Still, authors remarked that further experiments needed to be done before considering new function as a secure enough [5].

E. Design of S-boxes

By following Feistel recommendations, design of a new block cipher can be reduced to the design of S-boxes. That kind of design is not optimal, but is robust enough, and sufficient for many applications.

S-boxes or substitution boxes are generally n-input, m-output functions. Additionally, that can be viewed as a combination of m individual single output Boolean functions. In some applications, substitution boxes are formed by simple Boolean functions which take several Boolean inputs and give a single output.

There exists many ways to construct S-boxes. Generally, they can be divided the random way and wit the mathematical methods. Mathematical methods provide good cryptographic properties, but it can be vulnerable to algebraic attack if the expression is too simple. The random way can be to randomly generate S-boxes and test whether they are good. Another possibility is to construct S-boxes on the basis of some previously known S-boxes that have good properties [3] [7] [17].

a) Simulated Annealing for S-boxes

A cost function can be developed for single output Boolean function and generalize it for the use in S-boxes. Cost function represents the condition that needs to be satisfied, and that is the approach that authors adopted in [3].

Formal criteria for the single output Boolean functions are to have high nonlinearity and low autocorrelation. Those criteria are selected because they provide some level of protection against linear cryptanalysis, and differential cryptanalysis.

In year 2000, a new cost function family was proposed that offer significant improvement for the single output Boolean function case. That cost function defined the cost over the whole Walsh-Hadamard spectrum rather than on extreme values as it was done prior to that. The single output cost functions can be applied to each function defined as a linear combination of the outputs.

The search starts with regular, but randomly chosen function and moves around the search space. Simulated annealing was chosen to found the best solution. Annealing based search is used to minimize the value of the new cost function, and then hill-climb from the best solution with respect to nonlinearity to produce final solution. At the end, it is necessary to measure the nonlinearity, autocorrelation, and algebraic degree of the final solution.

The results obtained by this method are better then the results obtained in the case of human made S-boxes for the case when functions have small number of inputs. Furthermore, for some cases when number of inputs was larger, the results were also better then those obtained by human made S-boxes.

b) Genetic Algorithms for Bijective S-boxes

Authors constructed good bijective S-boxes that have high nonlinearity and low difference uniformity. The goal was to find good bijective S-boxes because they have been adopted by many algorithms. To achieve that goal, authors used genetic algorithms [6].

The solution candidates were represented by the truth tables and the fitness function is a function that takes into account nonlinearity and autocorrelation.

The genetic algorithm used can be described in the following way: first, a pool of randomly generated bijective S-boxes is generated. Each possible pair of S-boxes is selected for crossover (in paper called breeding). Apply mutation (hill climbing) on each of the offsprings. Compute the fitness of the offsprings and from a combined pool of parents and offsprings select best to create a new population.

Authors reported that with this strategy much stronger S-boxes can be obtained when compared to random ones. However, they remarked that additional research should be done to include avalanche effect and diffusion in the fitness function.

c) Genetic Algorithms for Self-inverse S-boxes

The main advantage of a self inverse S-box is that it needs less space in the implementation.

Authors continued their work from [6] which is presented in section above and in paper [7] they also considered self-inverse property of S-boxes. Additionally, to prevent "inbreeding"– mating of parents that are to similar they improved their selection strategy. As a way of preventing "inbreeding", an S-box distance was considered. An S-box distance describes the similarities between two S-boxes, i.e. the Hamming distance between all component output functions. The algorithm used is the same as described in section above, except for the differences mentioned. Based on the results obtained, authors concluded that it is possible to use genetic algorithms to evolve good self inverse S-boxes.

d) Optimal Tabu-genetic algorithm for S-boxes

Recently, a tabu-genetic algorithm for evolving S-boxes was presented [17].Author included a niche technique to maintain population diversity and to avoid premature convergence. Niche technique is used to enable evolutionary algorithm to find more then one optimal solution, but also to reduce redundant computations and fasten convergence. Author considered avalanche criteria and diffusion properties as evolution targets.

The Niche technique is added to the genetic algorithm to avoid trapping in the local optimum. When the convergence reaches the certain level, then tabu search as a local search mechanism is applied to converge to the global optimum.

The algorithm used can be described on the following way: create a pool of random solutions and find their fitness. Apply crossover and mutation to the individuals. Find the Hamming distance of every two individuals, compare the fitness degrees of those solutions and to the one with lower fitness give penalty function. Evaluate until optimal solution is obtained. Use the optimal solution from genetic algorithm for the initial solution for tabu search.

Results obtained by tabu-genetic algorithm showed that it was feasible and efficient way of creating S-boxes.

F. Design of Pseudorandom Sequence

It is possible to use one method from the area of evolutionary computation to evolve another method that can be utilized to reach certain objective. In their paper [4], authors presented a way of utilizing genetic algorithm to find cellular automata rules. Those set of rules are then used to make cellular automata to behave like a pseudorandom number generators that produce sufficiently random numbers for use in cryptography.

Although experiments with other computational intelligence methods in the making of pseudorandom number generators have been conducted, there are no good enough results for the use of those generators in field of cryptography.

Because of their simplicity, the cellular automata showed to be a good alternative for a pseudorandom number generation. Generators created on such a way, can also be easily implemented in the hardware.

A cellular automaton is a computational device composed of a uniform cell array and finite rules set that are applied to each cell. Many cellular automata exhibit a global chaotic and unpredictable behavior and, for that reason, they have been proposed to be used as pseudorandom sequence generators [4].

Authors used one-dimensional cellular automata with non-homogeneous local rules, in which each cell has between one and five arbitrary neighbors. Genetic algorithms were used as a way of generating good rules. The fitness function was computed based on statistical tests, and the entropy in cellular programming. The objective was to detect the rules that do not pass the tests.

The procedure was as follows: first, fitness function is calculated for every cell. The fitness of the cell is compared to the fitness of the neighbors and based on the results a mutation and crossover operations are performed. The rules that had best performance were used to generate pseudorandom sequences, and those sequences are compared with sequences obtained by other

pseudorandom number generators. The results done on a relatively small set of samples showed that cellular automata produced "more" random number than the Blum Blum Shub generator or the linear congruential generator, for example. Still, authors remarked that additional tests with a larger set of samples are needed to confirm the initial tests.

IV. CONLUSION

Evolutionary computation methods have been successfully applied to cryptology. However, it is necessary to notice that many of those implementations have been either to classical systems that have no real world application, or results obtained are difficult to reproduce and verify. More thorough research in the applications of state of the art cryptosystems is needed to be done if this area wants to shift to more significant part of cryptology. Applying more modern methods of evolutionary computation would probably yield better results in this area also.

REFERENCES

[1] J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda-Garnacho, and B. Ramos-Alvarez, "Wheedham: An Automatically Designed Block Cipher by means of Genetic Programming," IEEE Congress on Evolutionary Computation, CEC2006, Vancouver, pp. 192–199, 2006.

[2] A. J. Clark, "Optimisation Heuristics for Cryptology," PhD Thesis, Faculty of Information Technology, Queensland, 1998.

[3] J. A. Clark, J. L. Jacob, and S. Stepney, "The Design of S-Boxes by Simulated Annealing," IEEE Congress on Evolutionary Computation, CEC2004, Portland, vol 2., pp. 1533 – 1537, 2004.

[4] D. Delgado, D. Vidal, and G. Hernandez, "Evolutionary Design of Pseudorandom Sequence Generators based on Cellular Automata and Its Applicability in Current Cryptosystems," Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO '06, Seattle, pp. 1859-1860, 2006.

[5] J. M. Estevez - Tapiador, J. C. Hernandez - Castro, P. Peris-Lopez, and D A. Ribagorda, "Automated Design of Cryptographic Hash Schemes by Evolving Highly-Nonlinear Functions," Journal of Information Science and Engineering, vol. 24, pp. 1485-1504, 2008.

[6] C. Hua, and F. Deng-guo, "An Effective Evolutionary Strategy for Bijective S-boxes," IEEE Congress on Evolutionary Computation, CEC2004, Portland, vol 2., pp. 2120 - 2123, 2004.

[7] C. Hua, and F. Deng-guo, "An Effective Genetic Algorithm for Self-Inverse S-boxes," International Conference on Computational Intelligence and Security, Harbin, pp. 618-622, 2007.

[8] P. Isasi, and J. C. Hernandez, "Introduction to the Applications of Evolutionary Computation in Computer Security and Cryptography," Computational Intelligence, vol. 20, num. 3, pp. 445-449, 2004.

[9] J. Katz, and Y. Lindell, "Introduction to Modern Cryptography: Principles and Protocols," Chapman and Hall/CRC, USA, 2007.

[10] J. Koza, "Genetic Programming: On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)," The MIT Press, Cambridge, USA, 1992.

[11] M. Mitchell, "An Introduction to Genetic Algorithms," The MIT Press, Cambridge, USA, 1999.

[12] N. Nedjah and L. de Macedo Mourelle, "Multi-Objective Evolutionary Hardware for RSA-Based Cryptosystems," Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04), vol. 2, pp. 503-507, 2004.

[13] N. Nedjah, A. Abraham and L. de Macedo Mourelle (Eds.), "Computational Intelligence in Information Assurance and Security," in Studies in Computational Intelligence, Volume 57, Springer Berlin Heidelberg New York, 2007.

[14] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C," 2nd edition, Wiley; USA, 1996.

[15] A. Tragha, F. Omary, A. Mouloudi, "Genetic Algorithms Inspired Cryptography," A.M.S.E Association for the Advancement of Modeling & Simulation Techniques in Enterprises, Series D: Computer Science and Statistics, 2005.

[16] A. Tragha, F. Omary, A. Mouloudi, "ICIGA: Improved Cryptography Inspired by Genetic Algorithms," Proceedings of the International Conference on Hybrid Information Technology (ICHIT'06), pp. 335-341, 2006.

[17] Y. Xiangdong, "S Box Construction and Result Analysis Based on Optimal Tabu-genetic Algorithm," 2nd International Conference on Education Technology and Computer (ICETC), Shanghai , vol. 3, pp. 539-542, 2010.

[18] T. Weise, "Global Optimization Algorithms Theory and Application," 2009.