C^3PO: Cipher Construction with Cartesian genetic PrOgramming

Stjepan Picek
Delft University of Technology
s.picek@tudelft.nl

Karlo Knezevic
University of Zagreb, Faculty of Electrical Engineering and Computing
karlo.knezevic@fer.hr

Domagoj Jakobovic
University of Zagreb, Faculty of Electrical Engineering and Computing
domagoj.jakobovic@fer.hr

Ante Derek
University of Zagreb, Faculty of Electrical Engineering and Computing
ante.derek@fer.hr

ABSTRACT

In this paper, we ask a question whether evolutionary algorithms can evolve cryptographic algorithms when no precise design criteria are given. Our strategy utilizes Cartesian Genetic Programming in the bi-level optimization setting with multiple populations trying to evolve a cryptographic algorithm and break it. To challenge our design paradigm, we consider a number of scenarios with varying criteria on the system and its security. We are able to obtain interesting results in several scenarios where the attacker is not able to understand the text with more than a random chance. Interestingly, our system is able to develop various versions of one-time pads, which are the only systems that ensure perfect secrecy. Although our system is far from practical, we consider it interesting since it gives good results that are also human-readable.

CCS CONCEPTS

• **Security and privacy → Block and stream ciphers**; • **Computing methodologies → Discrete space search**;

KEYWORDS

Cryptography, Cartesian Genetic Programming, Block ciphers, Bi-level optimization

ACM Reference Format:
Stjepan Picek, Karlo Knezevic, Domagoj Jakobovic, and Ante Derek. 2019. C^3PO: Cipher Construction with Cartesian genetic PrOgramming. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3319619.3326869

1 INTRODUCTION

Cryptography is the science and engineering skill of designing cryptographic algorithms – ciphers. In symmetric key cryptography, all communicating parties use the same secret key [10]. The parties combine the message (plaintext) they want to exchange with the secret key through a series of transformations in order to obtain the encrypted message (ciphertext). Those transformations can be done only on the key, which is then a posteriori combined with the plaintext, or on both plaintext and the key at the same time. Such transformations enabling secure communication constitute a cryptographic algorithm, commonly known as a cipher. The process of designing a cipher is usually very complex since the designers need to follow a number of principles in order to create a secure cipher. In the design phase, it is necessary to consider both the properties of individual cipher components as well as the whole cipher. At the same time, that cipher needs to be tested against many possible attacks (e.g., differential cryptanalysis [2] or linear cryptanalysis [13]) in order to gain confidence in its strength. Although computers are extensively used in the design process to test certain parts of the cipher, the design of modern ciphers is mostly done by human experts.

In this paper, we pursue the goal of the automatic design of ciphers. In order to evolve ciphers, we use evolutionary algorithms, more precisely, Cartesian Genetic Programming (CGP) [17]. We believe such automatic cipher design is very interesting as 1) an exercise to explore what are the limits of evolutionary algorithms in modern cryptography and 2) a source of inspiration for new ciphers or their components. The automatic evolution of ciphers is a difficult task. That difficulty stems from the facts that we aim to develop a cipher that is easily usable by legitimate parties (commonly denoted as Alice and Bob [10]). At the same time, the malicious party (commonly known as Eve) should not be able to eavesdrop on that communication unless she has the key. In order to be as generic as possible (i.e., to allow the evolutionary algorithm to freely design a cipher), we impose no (or, as limited as possible) criteria on how the communication should happen. This constitutes a huge search space of solutions where 1) one side (we denote our cipher designer as Alice) generates a cipher and 2) Eve must not be able to understand the message since she does not have the key. Note, Eve knows both plaintexts and corresponding ciphertexts, which aims our setting into an attack model called the Known Plaintext Attack (KPA) [10]. Ideally, Eve would be able to use that information to develop some attack better than just random guessing. It is important to note that in the same way as we do not impose criteria in the evolution of ciphers, we also do not impose criteria on how Eve would attack the evolved ciphers.

Recently, Abadi and Anderson used two neural networks (Alice and Bob) to construct a cipher and a third network (Eve) to attack

GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326869

it, thus having adversarial environment between the first two networks and the third one [1]. The only constraints they imposed in the design process is that legitimate parties need to find a cipher such that they are able to communicate while Eve cannot decipher it. This should be possible to obtain since Eve has a much more difficult task because she does not know the secret key, while the legitimate parties know it. Abadi and Anderson were able to obtain interesting results, i.e., they found a way for Alice and Bob to communicate while Eve not being able to decipher that communication, but their approach has issues. While Eve was not able to decipher the communication with significantly better chances than random guessing (i.e., 50% of plaintext characters correctly guessed), Alice and Bob communicated "successfully" but their error was only somewhat lower than 50%. Next, since the cipher is a neural network architecture, no proper analysis of the design is actually possible. In the same way as it is difficult (impossible) to interpret the way Alice and Bob communicate, it is difficult to understand what is Eve doing so there is no guarantee the system is successful simply because Eve was not able to learn good attack strategies (i.e., the cipher could be insecure but Eve is just not able to break it).

Instead of neural networks, we use Cartesian Genetic Programming which is also a graph-based representation model; however, CGP solutions, when they form a sequence of instructions, are (potentially) understandable by human designers. We do not use the scenario where Alice and Bob independently try to develop the same cipher, since we do not see a practical justification for it. Indeed, it is sufficient that only one party generates a cipher and then shares it with all legitimate parties. Our setting uses bi-level optimization for Alice and Eve. Finally, since the design of a cipher also depends on its intended use, we set this as a constraint in our process. Indeed, the evolutionary algorithm (or any other technique) cannot guess how we might use the cipher. Consequently, we aim to design symmetric key ciphers where the difficulty of breaking them does not come from mathematically hard problems (like in the public key cryptography) but from the repetition of simpler operations. We are able to show our design strategy works over a number of different settings and produces solutions (i.e., ciphers) where the attacker best strategy is simply random guessing. At the same time, our solutions are short enough so it is possible to analyze them. Interestingly, even when not imposing any specific design constraints, our approach still finds some general paradigms of good cipher design (e.g., it is important to use the key – although this sounds trivial it is still a result obtained solely by the evolution process). Once we are able to automatically design a cipher, the question still remains why to use it. We emphasize that due to a lack of proper cryptanalysis, there must be a serious concern about the strength of such evolved ciphers that would prohibit them from being actually used.

The main contributions of this paper are:

- To the best of our knowledge, we are the first to consider such an open-ended cipher evolution process with evolutionary algorithms.
- We design a full system named C^3PO where Alice can design ciphers considering various criteria and cipher characteristics.
- We show that bi-level CGP is able to construct ciphers that Eve cannot break.

2 SYMMETRIC KEY CRYPTOGRAPHY

The plaintext P or message is the information that Alice and Bob wish to exchange. The encryption is a process of applying a transformation E to the plaintext P. After it, only an authorized party should be able to read the message. The ciphertext C is the result of encryption performed on plaintext using a cryptographic algorithm (cipher). The decryption is a process of applying a transformation D to the ciphertext C in order to obtain the plaintext.

A cryptographic algorithm (cipher) is a mathematical function used for encryption and decryption (ciphers are also used for other actions but those are outside the scope of this paper). From the attacker side, there are several models one can consider on the basis of an attacker's access to the system under attack. For instance, the attacker can have access only to the ciphertext (commonly known as Ciphertext-only attack). We can consider a more powerful attacker that has access to both ciphertexts and accompanying plaintexts. Such a model is called the Known Plaintext Attack (KPA) and is the model we assume Eve can use in this paper. Note, there are more powerful models than KPA but we consider them less relevant here since such models use specific attacks that Eve does not know. We note that a very powerful cryptanalysis technique called linear cryptanalysis is actually KPA.

Let us assume that Alice and Bob want to exchange some message they need to keep secret, i.e., that no one else can read it. They have only an insecure channel to communicate through. Alice could encrypt her message and send it encrypted over an insecure channel to Bob. If Bob has the same key as Alice, he can decrypt and read the message. At the same time, Eve cannot decrypt the message if she does not know the key. If Alice and Bob want to keep their communication private they need either to keep the key secret or the algorithm secret. Already in the 19th century, A. Kerchoff stated that a cryptosystem should be secure even if everything about the system, except the key, is publicly known [8].

For a computationally secure cryptosystem, C. Shannon deduced it should follow the confusion and diffusion principles [24]. The confusion principle means that the cipher output statistics should depend on the cipher input statistics in a manner too complicated to be exploited by the attacker. The confusion principle is related to the notion of nonlinearity since the attacker cannot easily approximate a cipher with a set of linear equations if the cipher possesses enough nonlinearity. More precisely, if a system is linear (i.e., S is a linear transformation) then $S(a) + S(b) = S(a + b)$. If S is a nonlinear transformation, $S(a) + S(b) \neq S(a + b)$ and in general, even if we know the result of $S(a)$ and $S(b)$, we do not know the result of $S(a + b)$. To measure the nonlinearity of a function, we need to measure its distance (e.g., the Hamming distance) to all linear and affine functions [3]. The diffusion principle relates to the fact that each digit of the input and each digit of the secret key should influence many digits of the output. This principle can be modeled through a general concept of avalanche criterion: a single bit change at the input must change at least half of the bits of the output (in the case exactly half of the bits must change then we talk about the strict avalanche criterion [26]).

Symmetric key cryptography can be divided into block and stream ciphers. The main differences between them is that given a message M and a ciphertext C when working with block ciphers it is hard to reconstruct the encryption transformation. When working with stream ciphers, the encryption transformation is easy and the security relies on the changing of that transformation for every symbol. One-time pad (OTP) is the only cryptographic system that ensures perfect secrecy, i.e., that no attacker can break it, provided that some rules are followed: the keys need to be 1) at least the same size as the plaintext, 2) random, 3) kept in secrecy, and 4) never reused.

3 RELATED WORK

Among the various heuristic techniques adopted for the problem of evolving Boolean functions (often used in stream ciphers) one can find *simulated annealing* [5], *genetic algorithms* [15], *genetic programming* and *Cartesian genetic programming* [18], *particle swarm optimization* [12], and *immunological algorithms* [21]. All those approaches follow the same line of reasoning: they define certain important cryptographic criteria that Boolean functions need to fulfill and they incorporate them in fitness functions. Next, researchers use heuristics to generate Substitution boxes (S-boxes) to be used in block ciphers. Examples use *simulated annealing with hill climbing* [4], *genetic algorithms* [14], *genetic programming* [19], *Cartesian genetic programming* [20], and *gradient descent method*[7]. Similar as for the Boolean functions, fitness functions contain specific properties that an S-box should possess. When considering the design of full ciphers, we can distinguish two options: in the first one, the design follows precise criteria defining the behavior of a cipher, while in the second direction, the process is more open-ended since there are no specific constraints. Examples in the first avenue encompass the design of pseudorandom number generators where the fitness function uses various types of randomness testing to evaluate whether the constructions offer sufficient randomness. The approaches use *genetic programming* [11] and *Cartesian genetic programming* [22]. A block cipher called Wheedham is designed by *genetic programming* where a fitness function ensures sufficient nonlinearity in the cipher [6].

Besides evolutionary computation, there are a number of papers belonging to the *neural cryptography* domain. There, a usual goal is to develop a key exchange protocol [23]. Still, such systems do not offer security as for instance shown by Klimov et al. [9]. Finally, adversarial neural networks are used to design a cipher where the only constraints are that Alice and Bob need to be able to exchange messages while Eve should not be able to eavesdrop on them [1]. As far as we are aware, this is the first attempt to build a whole cipher in such an "open" design style where Eve is also considered.

4 EXPERIMENTAL SETTING AND RESULTS

4.1 General Cipher Design Principles

As already stated in Section 1, our goal is the automatic design of ciphers where we do not impose criteria on how the cipher should be designed. Described as succinctly as possible, we want a cipher that legitimate parties can use to communicate and that the attacker cannot break. Everything else should (in an ideal case) be designed by the evolutionary process. It is not difficult to see that some additional questions need to be answered since evolutionary algorithms (EAs) cannot know the answers a priori.

(1) Who evolves our ciphers? In the evolution process, we do not use two parties (Alice and Bob) to evolve a cipher but only Alice. We see no reason to add additional work on the evolution process and make Bob guess what Alice finds. Correctly guessing the cipher is extremely difficult and just adds errors to legitimate parties communication. Additionally, there is no reason to limit communication to only two legitimate parties. In the system where one side develops a cipher and other sides need to guess it in order to communicate, adding more parties means even more error in the legitimate communication. Additionally, we do not want our cipher to be a secret, but only the key as in accordance with the Kerckhoff's principle, so Alice can develop a cipher and then simply send it over an insecure network.
(2) What kind of a cipher do we need, public key algorithm or symmetric key algorithm? In this paper, we consider only symmetric key algorithms.
(3) Does our algorithm use the key or not? Both options are possible: for instance, block/stream ciphers use keys while hash functions do not use them. We require that our designs use the key.
(4) Does the cipher operate on bits or blocks of bits? Although this seems like a detail that could be left to EA to decide, our experiments show that if we allow EA to operate on a bit level, it will not group those bits into blocks. We decide to work with blocks and consequently, to design block ciphers.
(5) What is the size of plaintext, ciphertext, and key? We consider a scenario where all have the same size and that is either 4 bits or 8 bits.

Note, all decisions given above are our design choices and not something devised due to some constraints on EA. Indeed, we could have decided to work on stream ciphers or hash functions and EA should still be able to produce results. While 4 or 8 bits can look too small a size to be practical, it is easy to apply the same cipher to any number of bits in parallel and arrive at more practical sizes. Of course, since we consider each block separately, there is no diffusion between the blocks, but we do not consider this as a problem at this phase. Finally, although it is common for block ciphers to be iterated [10], i.e., to operate in a number of rounds in order to improve their security, we consider only a single round. Trivially, each new round adds a certain amount of security to the cipher but we measure here the security only with respect to Eve. Then, if Eve cannot break a single round cipher, there is no reason to add more rounds since Eve will not be able to break them. If Eve is able to break a single round, then adding rounds would make the attacks more difficult. Still, our experiments show there are many designs with a single round strong enough so Eve cannot break them. Consequently, we see no need to consider multiple rounds at this point.

Besides these general constraints, our experiments indicated arguments for several additional constraints that are more specific. If we require our cipher to be bijective, i.e., invertible, we need to explicitly encode this constraint. This may seem like a limitation but is actually logical since EA does not know whether there is another side (or sides) that need to be able to decrypt the ciphertext by inverting the encryption procedure. We note this is not an absolute requirement since any function can become invertible if used with the Feistel structure [10].

Once the evolution process is finished and we have the cipher that Eve was not able to break, we would require some assurance that our cipher is strong. If Eve cannot break the cipher, there are two possible extremes: 1) the cipher is strong and that is the reason Eve cannot break it, or 2) Eve did not learn any good way to analyze the cipher, so although the cipher is weak, she didn't break it. In order to gain some assurance that the cipher is strong and that Eve is a capable attacker, we measure the confusion and diffusion as given by nonlinearity and avalanche criterion, respectively.

4.2 Cartesian Genetic Programming

In Cartesian Genetic Programming (CGP), a program is represented as an indexed graph. The terminal set (inputs) and node outputs are numbered sequentially. Node functions are also numbered separately. CGP has three parameters to be chosen by the user: number of rows n_r, number of columns n_c, and levels-back l [16].

In our experiments, for the number of rows we use a value of one and for the levels-back parameter, we use the same value as for the number of columns. Therefore, the maximal number of nodes in CGP is equal to the number of columns. The number of node input connections n_n is two and the number of node output connections n_o is one. A maximal node arity is two. The function set in all experiments is AND, OR, NOT, XOR, ROR, and ROL (rotation one bit right and left). All functions are vectorial, which means their arguments are vectors of the same size as the result vector. All functions operate on the bit level, i.e., AND function is defined as $AND(\vec{x}, \vec{y}) = (x_0 \text{ AND } y_0, ..., x_n \text{ AND } y_n)$. Each CGP genotype has two inputs and a single output. Depending on a model, inputs are fed by vectors of plaintext and key or plaintext and ciphertext, and output vector is ciphertext or the guessed secret key.

The population size for CGP equals five in all our experiments. For CGP individual selection, we use a (1 + 4) selection strategy in which offspring are favored over the parent when they have a fitness less than or equal to the fitness of the parent. We use a single active gene mutation where individual genes are mutated until an active gene (i.e., a gene contributing to at least one output node) is affected.

4.2.1 Bi-level optimization. Bi-level optimization is a special kind of optimization where one problem is embedded within another one [25]. The outer optimization task is commonly referred to as the upper-level optimization task, and the inner optimization task is commonly referred to as the lower-level optimization task. In our case, Alice does the upper-level optimization task referring to Eve's lower-level task.

When evaluating each individual in the upper-level population (each Alice), a new lower-level population (of Eves) is created. The chosen Alice individual's cipher is used to generate training set pairs of plaintext and ciphertext; this training set is used at the lower level to evaluate lower-level individuals. After the lower-level evolution is terminated, the best solution from the lower level is used to estimate the fitness of Alice, i.e., the upper-level individual.

The same evolutionary algorithm and the same representation is used at both levels, but the fitness functions and termination criteria are different. In our case, the size of the lower-level population is $\sigma = 3$, and the lower-level termination criterion is the number of evaluations. This process is illustrated in Algorithm 1.

Algorithm 1 Alice and Eve evaluation by bi-level optimization.

```
Input: $\mathbb{X}_{train} = \{P_i, K_i\}^N$ – training dataset, $\mathbb{X}_{test} = \{P'_i, K'_i\}^M$ – test dataset, $I$ – iterations
$\pi = 0$
repeat
    $C$ = encrypt ($\mathbb{X}_{train}$, $Alice$)
    $\Pi^{\sigma}_{Eve} = Eve_1, ..., Eve_{\sigma}$
    for $\forall Eve$ in $\Pi^{\sigma}_{Eve}$ do
        $Eve_i$ = build model ($\{P, C\}^N$)
    end for
    $Eve$ = best model in $\Pi^{\sigma}_{Eve}$
    $C'$ = encrypt ($\mathbb{X}_{test}$, $Alice$)
    $\hat{K'}$ = find secret key ($\{P', C'\}^M$, $Eve$)
    calculate $cost_{Alice}$ based on $\hat{K'}$ and $K'$
    $inc(\pi)$
until $\pi < I$
```

Table 1: Parameters for CGP.

Parameter	Value
Genotype length	20/40 nodes (4/8 bits)
Input/Output nodes	2/1
Evolutionary selection	(1+4)
Mutation type	single active gene
Function set	AND, OR, NOT, XOR, ROR, ROL
Maximum evaluations	25 000/50 000 (Alice/Eve)
Runs per experiment	30

4.3 Common Parameters and Datasets

All CGP parameters are given in Table 1. In all experiments, the number of independent trials for each configuration is 30. We use two datasets depending on the message length that can be $n = 4$ and $n = 8$ bits. Here, both the message and the secret key are of the same length. The message and the secret key consist of uniformly distributed binary values. After a tuning phase, we set the number of nodes in CGP for 4-bit messages to 20 and for 8-bit messages to 40 nodes. To encrypt a message block consisting of n bits with a key consisting of n bits as well, we use vectorial functions that produce n bits of ciphertext. Consequently, Alice consists of 2 input nodes and 1 output node. Eve has information about the pairs of plaintext and ciphertext so she also has 2 input nodes and 1 output node, where output represents the secret key used in the encryption. Our encryption algorithm always outputs ciphertexts of the same size as is the plaintext. For the 4-bit messages, the training dataset contains $P = 150$ messages and $K = 10$ keys ($N = 150$, training set size), while testing dataset contains $P' = 50$ messages and $K' = 6$ keys ($M = 50$, testing set size). For the 8-bit messages, the training dataset contains $P = 400$ messages and $K = 50$ keys ($N = 400$) while testing dataset contains $P' = 100$ messages and $K' = 10$ keys ($M = 100$). All pairs plaintext/key are selected uniformly at random. We divide our evolution process into a number of evaluations E and iterations I. A single iteration I represents a process where all parties undergo E evaluations. Training and testing set is regenerated after each iteration. In all our experiments, the evolution does $I = 50$ iterations. We set the number of evaluations E to 25 000 for Alice and 50 000 for Eve. By performing a larger number of evaluations for Eve, we give her an advantage over Alice in order to build as powerful attack as possible, which in turn helps evolving stronger ciphers. In total, each experimental run has at least 1 250 000 evaluations for each of the populations.

4.4 Cost Functions

We use the L1 distance to measure the difference between the actual key K and the guessed key $\hat{K}'$. The L1 distance is defined as $d\left(K, \hat{K}'\right) = \sum_{i=1}^{n} |K_i - \hat{K}'_i|$, where n equals the message/key length. Alice's $cost_A$ expression represents Eve's error where the global optimum is reached when half (on average) of the decrypted message bits are wrong. If Eve is correct in only half the bits, that essentially means she is random guessing (independent coin toss for each message bit), which is the optimal scenario from Alice's perspective. Note, if Eve would be wrong in significantly more than half the bits, we could simply invert her guesses (swap all 0s for 1s and vice versa), which would make her wrong in significantly less than half of the bits. The second component $crypto_{Eve}^{Alice}$ describes the simultaneous fulfillment of cryptographic properties. Alice's cost function is measured by Eve and then, in a bi-level optimization sent to Alice.

$$cost_{Alice} = \left|\frac{n}{2} - d(K, \hat{K}'_{Eve})\right| + crypto_{Eve}^{Alice}. \quad (1)$$

$$cost_{Eve} = min(d(K, \hat{K}'_{Eve}), n - d(K, \hat{K}'_{Eve})). \quad (2)$$

$$crypto_{Eve}^{Alice} = \widehat{NL} + \widehat{diff} + bijectivity. \quad (3)$$

The bijectivity property equals 0 if Alice constructs a non-bijective cipher and 1 otherwise. The nonlinearity NL and diffusion $diff$ are statistically measured with a subset of secret keys from the training set. To reduce the NL and $diff$ time complexity, we randomly choose 3 keys and calculate cipher's nonlinearity and diffusion, where $\widehat{NL} = min\{NL_{key^i}\}$ and $\widehat{diff} = min\{diff_{key^i}\}$. Since the parameters values have different co-domains, we scale $\widehat{NL}$ and $\widehat{diff}$ to the interval $[0, n]$, using transformation $scaled = n\frac{optimal-value}{optimal}$. The best possible nonlinearity value for 4 bit messages is 4 and for 8 bit messages is 112. The highest diffusion for each bit of the output is equal to $\frac{n^2}{2}$ because, for each message, there exist n messages differing in only 1 bit, which should have cipher difference in $\frac{n}{2}$ bits. Since we are minimizing our fitness functions, the obtained nonlinearity and diffusion are subtracted from the best possible values.

5 RESULTS

We divide our experiments into five scenarios with respect to the cryptographic cost from Alice's perspective. The first scenario considers Eq. (1) with Eq. (3) set to 0, i.e., a setting where Alice tries to evolve a system where Eve would make a mistake on as close to the half of bits as possible. In the second scenario, we additionally consider Alice's diffusion, in the third scenario nonlinearity, and in the fourth scenario bijectivity. Finally, we combine the second, third, and fourth scenario into the fifth scenario. The results for all scenarios are given in Tables 2 and 3, and Figures 1a–3b. The best obtained values over all scenarios are given in bold style. Note that in Table 3, the value *keys broken* we aim to maximize and *number of wrong key bits* we aim to make as different from $n/2$ as possible (since for both of those columns we consider the situation from Eve's perspective). In Tables 2 and 3, there are two columns for $cost_{Eve}$ that show different results. In Table 2, we give $cost_{Eve}$ as measured in upper-level optimization from Alice, while in Table 3, we give $cost_{Eve}$ as measured in lower-level optimization from Eve. For the column *active nodes*, we do not give any values in bold style since we do not optimize the size of the network nor we assume that one size is better than some other (in general, if evolved cipher/Eve's attack are strong, then as small number of active nodes as possible would potentially ease human interpretability of the results).

5.1 Scenario 1

In this scenario, $crypto_{Eve}^{Alice}$ is set to 0 in Eq. (1) and only Eve's L1 measure is optimized. For Eve, the aim is to guess as many bits as possible (or as few bits as possible, see Eq. (2)). Then, the best solution from Eve's lower-level population is used to calculate the fitness for the current Alice candidate. In Table 2, when considering 4-bit setting, Alice evolves a cipher that is small (as can be observed by the number of active nodes) but the cipher is relatively weak since Eve is able to guess the most bits from all scenarios. Next, we see that both diffusion and nonlinearity are bad. This indicates that our ciphers are not providing a lot of security. Additionally, we can observe that the evolved ciphers are most of the time not bijective.

The setting with 8 bits displays a similar behavior, where values are slightly larger but also the range of obtainable values is much larger. At the same time, Eve's cost shows only a small increase from 4-bit setting to 8-bit setting when compared with Alice. Still, Eve is again for 8-bit setting the least successful when considering Scenario 1. This means that Eve is able to perform significantly better than the random guess. This fact coupled with a relatively small increase in the number of active nodes suggests that for 8-bit setting, Alice would require longer evolution process in order to evolve strong ciphers.

In Table 2, we show the situation after Eve's evaluation where we see that she is able to break the largest number of keys for 4-bit setting but the number of wrong key bits is close to $n/2$, which indicates that most of the time she is not much better than the random guess.

For 8-bit setting, we see that the number of broken keys is smaller than for 4-bit setting but that she deviates slightly more from the random guess performance. This suggests that Eve needs longer evolution process to cope with the larger key/message size.

5.2 Scenario 2

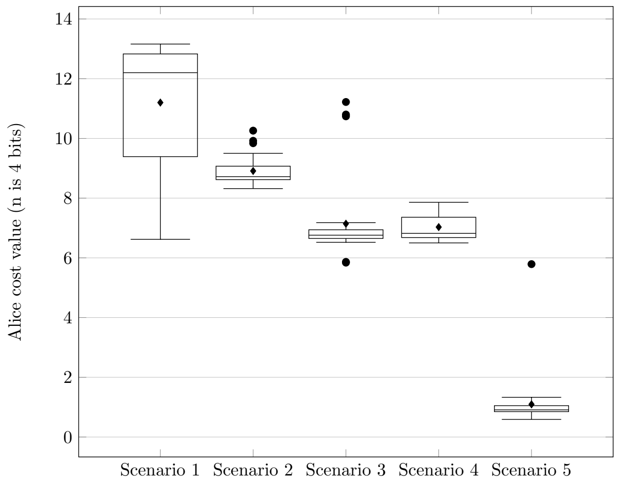
In this scenario, besides Eve's L1 measure, the fitness component $crypto_{Eve}^{Alice}$ includes the diffusion. When considering Alice's performance as given in Table 2, we see a significant change when compared with Scenario 1. Alice's cost is reduced, which means she is developing better ciphers (i.e., those that are more difficult for Eve) but she also needs more active nodes on average. Since we optimize for diffusion, we see that for both 4 and 8 bits, the diffusion significantly improves, which also improves the value for $crypto_{Eve}^{Alice}$. The ciphers are never bijective (both for 4 and 8 bits) and the nonlinearity decreases. This indicates that if we optimize for

Table 2: Alice, average results for all scenarios.

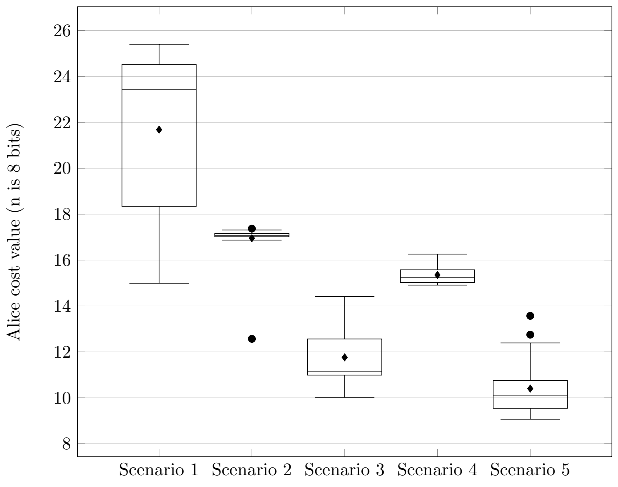
N	Scenario	average ± standard deviation						
		$cost_{Alice}$	$cost_{Eve}$	$crypto_{Eve}^{Alice}$	active nodes	diffusion	nonlinearity	bijectivity
4 bits	1	11.20 ± 2.43	0.81 ± 0.17	10.38 ± 2.40	4.60 ± 2.36	6.63 ± 1.56	3.60 ± 1.20	0.13 ± 0.33
	2	8.91 ± 0.46	0.76 ± 0.24	8.15 ± 0.39	5.16 ± 2.33	**0.30 ± 0.78**	4.00 ± 0.00	0.00 ± 0.00
	3	7.14 ± 1.29	0.78 ± 0.15	6.35 ± 1.29	5.30 ± 1.91	3.91 ± 0.51	0.40 ± 1.20	0.00 ± 0.00
	4	7.03 ± 0.46	0.76 ± 0.13	6.26 ± 0.44	3.23 ± 2.01	4.53 ± 0.88	4.00 ± 0.00	**1.00 ± 0.00**
	5	**1.09 ± 0.88**	**0.68 ± 0.16**	**0.41 ± 0.89**	10.00 ± 1.50	0.56 ± 0.36	**0.14 ± 0.71**	**1.00 ± 0.00**
8 bits	1	21.68 ± 3.78	1.11 ± 0.15	20.56 ± 3.76	7.73 ± 3.35	26.48 ± 4.08	105.60 ± 19.20	0.20 ± 0.40
	2	16.95 ± 0.82	1.08 ± 0.11	15.86 ± 0.82	9.96 ± 2.40	**0.07 ± 0.36**	109.87 ± 11.48	0.00 ± 0.00
	3	11.76 ± 1.37	1.08 ± 0.11	10.67 ± 1.36	15.20 ± 2.48	9.04 ± 3.62	**5.87 ± 7.71**	0.00 ± 0.00
	4	15.35 ± 0.32	1.10 ± 0.10	14.25 ± 0.28	5.70 ± 3.11	25.00 ± 1.12	112.00 ± 0.00	**1.00 ± 0.00**
	5	**10.40 ± 1.13**	**1.06 ± 0.11**	**9.33 ± 1.12**	13.16 ± 2.92	5.34 ± 4.46	112.00 ± 0.00	**1.00 ± 0.00**

Table 3: Eve, average results for all scenarios.

N	Scenario	average ± standard deviation			
		$cost_{Eve}$	active nodes	keys broken	number of wrong key bits
4 bits	1	**0.78 ± 0.51**	7.40 ± 3.08	**4.20 ± 1.83**	1.97 ± 1.14
	2	1.23 ± 0.21	8.10 ± 2.79	2.80 ± 1.72	2.02 ± 0.26
	3	1.14 ± 0.34	7.93 ± 2.93	2.63 ± 1.68	1.91 ± 0.59
	4	0.86 ± 0.54	6.96 ± 2.62	3.56 ± 2.06	**1.80 ± 1.03**
	5	1.17 ± 0.33	6.63 ± 2.18	2.40 ± 1.94	1.96 ± 0.22
8 bits	1	1.98 ± 1.05	14.16 ± 6.58	3.76 ± 3.66	4.14 ± 2.09
	2	2.66 ± 0.72	17.26 ± 4.28	1.66 ± 2.64	**3.67 ± 1.01**
	3	2.82 ± 0.20	17.50 ± 3.98	1.10 ± 1.42	3.96 ± 0.42
	4	**1.62 ± 1.39**	12.36 ± 5.55	**5.06 ± 4.48**	4.00 ± 2.63
	5	2.77 ± 0.32	18.60 ± 3.65	1.53 ± 1.82	3.81 ± 0.72



(a) $n = 4$.



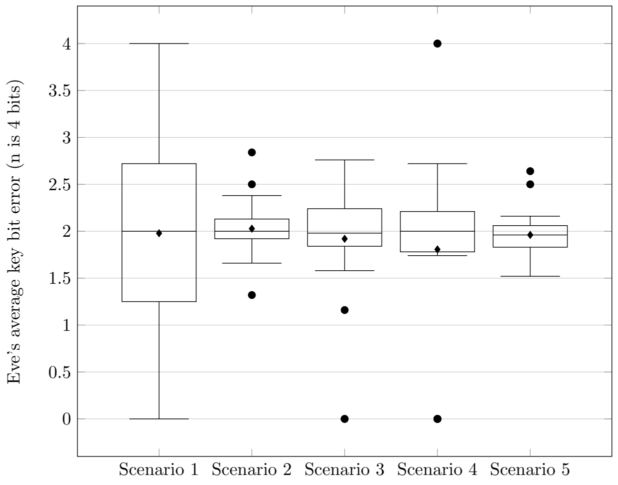
(b) $n = 8$.

Figure 1: Alice's cost function values.

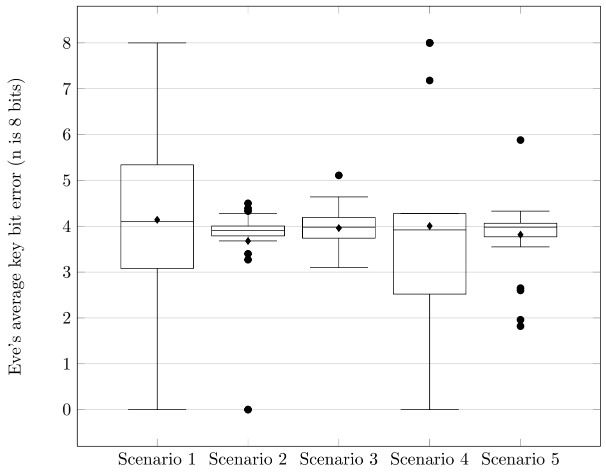
diffusion, we lose on nonlinearity. Considering Eve's performance in Table 3, we see that the cost increases, which means this scenario is more difficult for her. She also needs more active nodes, is able to break fewer keys, and is closer to the random guess behavior than for Scenario 1.

5.3 Scenario 3

This scenario considers Eve's L1 measure and nonlinearity in the $crypto_{Eve}^{Alice}$ term. In Table 2, we observe that as expected, nonlinearity is significantly better than in the previous scenarios. At the same time, apart from the diffusion, the other components are largely not affected. As in the previous cases, bijectivity is not obtained, which again indicates that this property is not intrinsic to

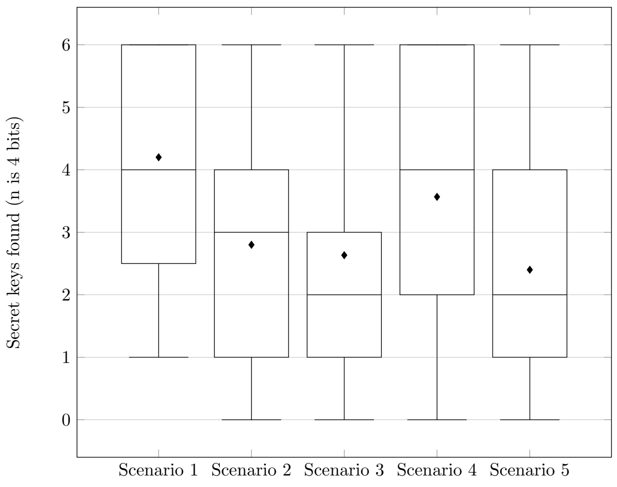


(a) $n = 4$.

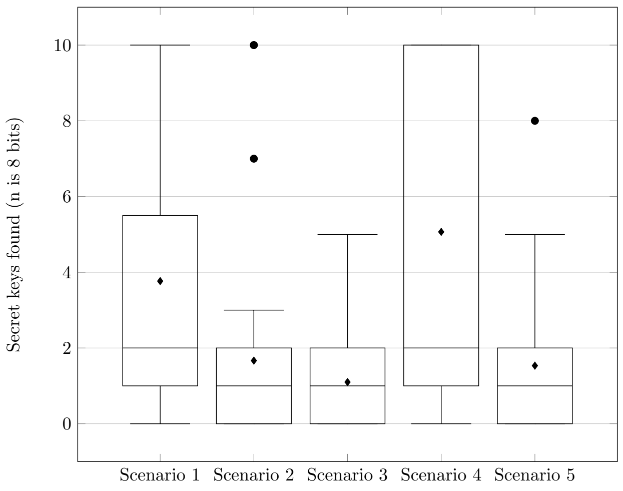


(b) $n = 8$.

Figure 2: Eve's average key bits error.



(a) $n = 4$.



(b) $n = 8$.

Figure 3: Eve's found keys in test set.

the design of Alice's cipher. The sizes for ciphers that Alice evolves are very similar for Scenario 2 and 3, which means that the properties are similar in complexity to add to the cipher design. When considering the results from Alice's perspective, it is difficult to say whether Eve has more problems for Scenario 2 or Scenario 3 since the results are very similar. Still, Alice's cost is lower than for Scenario 2, which suggests that high nonlinearity contributes more to strong ciphers than high diffusion. When considering the results from Eve/s perspective (Table 3), we see that for 4-bit setting she is able to improve the results when compared to Scenario 2, but for 8-bit setting her results are worse, i.e., the cost increases. For both 4-bit and 8-bit setting, the number of keys broken decreases when compared to Scenarios 1 and 2. For the 4-bit setting, the number of the wrong key bits shows she is moving away from random guessing while the situation for 8-bit setting is opposite.

5.4 Scenario 4

This scenario considers the bijectivity property while optimizing Eve's L1 measure. As it can be seen in Table 2, when explicitly included as a criterion, CGP can easily adapt and provide a bijective cipher. It is important to remember that the bijectivity is not tested and guaranteed on the whole domain, but only on the generated plaintext dataset. Interestingly, in 4-bit setting, making a cipher bijective is still causing similar problems for Eve despite the fact that diffusion and nonlinearity are not as good as for Scenarios 2 and 3. From the evolved solutions perspective, this scenario requires the least number of active nodes among all considered scenarios.

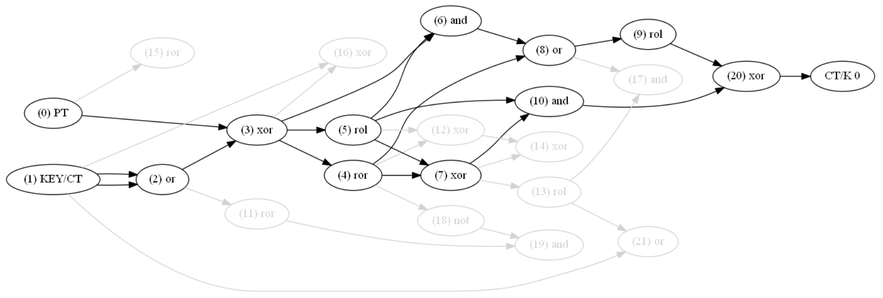


Figure 4: Example of an evolved cipher obtained by Alice.

From Table 3, we observe that Eve does not require many active nodes to reach the second best result for 4-bit setting and the best result for 8-bit setting. For 8-bit setting, she is able to break the most keys from all considered scenarios. Interestingly, for 4-bit setting, we see that she is the furthest from random guessing while for 8-bit setting, she is on average random guessing. Still, the highest deviation value in this case indicates she is able to either display excellent behavior or she simply gets stuck and cannot do anything better than the random guess. The number of active nodes suggests that Eve is able to do good attacks with relatively small networks (which, in turn, suggests that Alice is probably not constructing strong ciphers).

5.5 Scenario 5

In this scenario, we optimize Eve's L1 measure, as well as nonlinearity, diffusion, and bijectivity. From Table 2, we see that evolved ciphers are bijective and with good diffusion, although not as good as in Scenario 2. When considering 4-bit setting, we see we are also able to obtain the best possible nonlinearity while for 8 bits we did not obtain nonlinear functions. This means that the problem is simply too difficult or that the evolution process was too short when considering all conditions. Still, even with such differing results for those two settings, we see Alice's cost is the smallest from all considered scenarios and that Eve has the biggest difficulties there. In fact, when considering the results from Alice's perspective and Eqs. (1), (2), and (3), we see that for both 4-bit and 8-bit setting, the results are the best from all considered scenarios.

When considering this scenario from Eve's perspective as given in Table 3, Eve's cost is the second highest for both 4-bit and 8-bit settings, which means that there are scenarios more difficult for Eve (those considering only diffusion and only nonlinearity, respectively). Interestingly, for the 4-bit setting, Eve uses the least active nodes while for the 8-bit scenario she uses the most active nodes. This indicates that she is somewhat "deceived" for the 4-bit case to think the problem is easy. Both the number of the keys broken and the number of wrong key bits indicate Scenario 5 not to be the most difficult one for Eve.

5.6 General Observations

We consider Scenario 5 to be the best for Alice (and consequently the worst for Eve) since all statistical indicators for cryptographic properties are balanced, which can be seen in Table 2. Additionally, Eve is making mistakes in many bits, which means that this scenario is difficult for her. Interestingly, while our results clearly show that Scenario 5 is the best for Alice, it is not straightforward to conclude it is the worst for Eve. Indeed, Table 3 clearly shows the only instance where Scenario 5 is the worst one for Eve is for 4-bit setting when considering the number of keys broken.

For Scenario 1, where evolution has the least guidance, the results are still not too bad from Alice's perspective. Still, a high standard deviation value tells us that the quality of evolved ciphers differs significantly.

In Figure 1, we depict the results for all 5 scenarios, 4-bit and 8-bit setting. There, it is easy to see that the first scenario is the worst one, scenarios 2–4 are similar, and Scenario 5 is the best one. It is interesting to note that in several scenarios, we obtained the one-time pad cipher and its variants, which is the only cipher providing the perfect secrecy.

In Figure 2, we depict Eve's average key bit errors and we can see that while the values are indeed around $n/2$, which indicate random guessing, Scenario 1 has $\approx \pm 1$ from $n/2$ while other scenarios differ less from that value. Figure 3 depicts the number of keys broken. For 4-bit setting, the results are more similar with a small advantage for Scenario 3, while for 8-bit setting, Scenarios 2, 3, and 5 exhibit similar behavior. Finally, we depict one cipher operating on 4 bits evolved by Alice in Figure 4.

6 CONCLUSIONS

In this paper, we investigate how to automatically evolve ciphers with CGP and bi-level optimization. Our results show that we are able to develop ciphers that are (relatively) resilient against Eve's attacks and that use only a small number of active nodes, which makes them easier to interpret. Once we add more properties that a cipher needs to fulfill, the results are naturally improved. Eve is not able to be significantly more successful than if she would be random guessing. We consider this to be only a proof of a concept that shows EA has potential as an automatic cipher builder. Naturally, to obtain something useful in practice, more experiments and improvements are necessary.

In future work, we will consider more variations in the number of evaluations and/or size of CGP graph. Besides that, we notice that our cost function is often too strict, which results in getting stuck in local optima. To remedy that, we aim to design cost functions that gradually add constraints during the evolution process and do not impose all of them from the beginning of the evolution process.

REFERENCES

[1] Martín Abadi and David G. Andersen. 2016. Learning to Protect Communications with Adversarial Neural Cryptography. *CoRR* abs/1610.06918 (2016). arXiv:1610.06918 http://arxiv.org/abs/1610.06918

[2] Eli Biham and Adi Shamir. 1991. Differential Cryptanalysis of DES-like Cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '90)*. Springer-Verlag, London, UK, 2–21. http://dl.acm.org/citation.cfm?id=646755.705229

[3] Claude Carlet. 2010. Vectorial Boolean Functions for Cryptography. In *Boolean Models and Methods in Mathematics, Computer Science, and Engineering* (1st ed.), Yves Crama and Peter L. Hammer (Eds.). Cambridge University Press, New York, USA, 398–469.

[4] John A. Clark, Jeremy L. Jacob, and Susan Stepney. 2005. The design of S-boxes by simulated annealing. *New Generation Computing* 23, 3 (Sept. 2005), 219–231. https://doi.org/10.1007/BF03037656

[5] John A. Clark, Jeremy L. Jacob, Susan Stepney, Subhamoy Maitra, and William Millan. 2002. Evolving Boolean Functions Satisfying Multiple Criteria. In *INDOCRYPT 2002 (LNCS)*, Alfred Menezes and Palash Sarkar (Eds.), Vol. 2551. Springer, 246–259.

[6] J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda-Garnacho, and B. Ramos-Alvarez. 2006. Wheedham: An Automatically Designed Block Cipher by means of Genetic Programming. In *2006 IEEE International Conference on Evolutionary Computation*. 192–199. https://doi.org/10.1109/CEC.2006.1688308

[7] Oleksandr Kazymyrov, Valentyna Kazymyrova, and Roman Oliynykov. 2013. A Method For Generation Of High-Nonlinear S-Boxes Based On Gradient Descent. Cryptology ePrint Archive, Report 2013/578. (2013).

[8] A. Kerckhoffs. 1883. La cryptographie militaire. *Journal des Sciences Militaires* (1883), 161–191.

[9] Alexander Klimov, Anton Mityagin, and Adi Shamir. 2002. Analysis of Neural Cryptography. In *Advances in Cryptology — ASIACRYPT 2002*, Yuliang Zheng (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 288–298.

[10] Lars R. Knudsen and Matthew Robshaw. 2011. *The Block Cipher Companion.* Springer. I–XIV, 1–267 pages.

[11] Carlos Lamenca-Martinez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda. 2006. Lamar: A New Pseudorandom Number Generator Evolved by Means of Genetic Programming. In *Parallel Problem Solving from Nature - PPSN IX*. Springer Berlin Heidelberg, Berlin, Heidelberg, 850–859.

[12] Luca Mariot and Alberto Leporati. 2015. Heuristic Search by Particle Swarm Optimization of Boolean Functions for Cryptographic Applications. In *GECCO Companion '15*. ACM, 1425–1426.

[13] Mitsuru Matsui and Atsuhiro Yamagishi. 1993. A new method for known plaintext attack of FEAL cipher. In *Proceedings of the 11th annual international conference on Theory and application of cryptographic techniques (EUROCRYPT'92)*. Springer-Verlag, Berlin, Heidelberg, 81–91. http://dl.acm.org/citation.cfm?id=1754948.1754958

[14] W. Millan, L. Burnett, G. Carter, A. Clark, and E. Dawson. 1999. Evolutionary Heuristics for Finding Cryptographically Strong S-Boxes. In *Information and Communication Security*. LNCS, Vol. 1726. Springer Berlin Heidelberg, 263–274.

[15] William Millan, Andrew Clark, and Ed Dawson. 1998. Heuristic Design of Cryptographically Strong Balanced Boolean Functions. In *EUROCRYPT '98*. 489–499.

[16] Julian F. Miller (Ed.). 2011. *Cartesian Genetic Programming*. Springer Berlin Heidelberg.

[17] Julian F. Miller and Peter Thomson. 2000. Cartesian Genetic Programming. In *EuroGP*. 121–132.

[18] Stjepan Picek, Domagoj Jakobovic, Julian F. Miller, Elena Marchiori, and Lejla Batina. 2015. Evolutionary Methods for the Construction of Cryptographic Boolean Functions. In *Genetic Programming - 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*. 192–204.

[19] Stjepan Picek, Luca Mariot, Bohan Yang, Domagoj Jakobovic, and Nele Mentens. 2017. Design of S-boxes Defined with Cellular Automata Rules. In *Proceedings of the Computing Frontiers Conference (CF'17)*. ACM, New York, NY, USA, 409–414. https://doi.org/10.1145/3075564.3079069

[20] Stjepan Picek, Julian F. Miller, Domagoj Jakobovic, and Lejla Batina. 2015. Cartesian Genetic Programming Approach for Generating Substitution Boxes of Different Sizes. In *GECCO Companion '15*. ACM, New York, NY, USA, 1457–1458.

[21] Stjepan Picek, Dominik Sisejkovic, and Domagoj Jakobovic. 2017. Immunological algorithms paradigm for construction of Boolean functions with good cryptographic properties. *Eng. Appl. of AI* 62 (2017), 320–330. https://doi.org/10.1016/j.engappai.2016.11.002

[22] Stjepan Picek, Dominik Sisejkovic, Vladimir Rozic, Bohan Yang, Domagoj Jakobovic, and Nele Mentens. 2016. Evolving Cryptographic Pseudorandom Number Generators. In *Parallel Problem Solving from Nature – PPSN XIV*. Springer International Publishing, Cham, 613–622.

[23] A. Ruttor. 2007. *Neural Synchronization and Cryptography*. Ph.D. Dissertation. PhD Thesis, 2007.

[24] C.E. Shannon. 1949. Communication theory of secrecy systems. *Bell System Technical Journal* 28, 4 (1949), 656–715.

[25] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. 2018. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation* 22, 2 (2018), 276–295.

[26] A. F. Webster and S. E. Tavares. 1986. On the Design of S-Boxes. In *Advances in Cryptology — CRYPTO '85 Proceedings*, Hugh C. Williams (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 523–534.