Evolutionary Algorithms for Boolean Functions in Diverse Domains of Cryptography

Stjepan Picek stjepan@computer.org
KU Leuven, ESAT/COSIC and iMinds, Kasteelpark Arenberg 10, bus 2452, B-3001 Leuven-Heverlee, Belgium and
LAGA, UMR 7539, CNRS, University of Paris 8, France

Claude Carlet claude.carlet@gmail.com
LAGA, UMR 7539, CNRS, University of Paris 13 and University of Paris 8, France

Sylvain Guilley sylvain.guilley@telecom-paristech.fr
TELECOM-ParisTech, Paris, France & Secure-IC S.A.S., Rennes, France

Julian Miller julian.miller@york.ac.uk
Department of Electronics, University of York, York, UK

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

Abstract

The role of Boolean functions is prominent in several areas like cryptography, sequences, and coding theory. Therefore, various methods for the construction of Boolean functions with desired properties are of direct interest. New motivations on the role of Boolean functions in cryptography with attendant new properties have emerged during the years. There are still many combinations of design criteria left unexplored and in this matter evolutionary computation can play a distinct role. This paper concentrates on two scenarios for use of Boolean functions in cryptography. The first uses Boolean functions as the source of the nonlinearity in filter and combiner generators. Although relatively well explored using evolutionary algorithms, it still presents an interesting goal in terms of the practical sizes of Boolean functions. The second scenario appeared rather recently where the objective is to find Boolean functions that have various orders of the correlation immunity and minimal Hamming weight. In both those scenarios we see that evolutionary algorithms are able to find high quality solutions where genetic programming performs the best.

Keywords

Evolutionary algorithms, Boolean functions, Cryptography, Comparison.

1 Introduction

In cryptography, one standard division is into symmetric key cryptography and public key cryptography (Diffie and Hellman, 1976; Paar and Pelzl, 2010). Going one step further, symmetric key cryptography can be again divided into block ciphers and stream ciphers. A common trait for all these ciphers is that they are designed in accordance with the number of cryptographic criteria they need to fulfill. These varying criteria enable ciphers to resist various cryptanalysis attacks. Some of the most common attacks on block ciphers are differential (Biham and Shamir, 1991) and linear (Matsui and

Yamagishi, 1993) cryptanalysis. While on stream ciphers, attacks are the Berlekamp-Massey (Massey, 1969), fast correlation (Meier and Staffelbach, 1988), algebraic (Courtois and Meier, 2003), and fast algebraic (Courtois, 2003) cryptanalysis, and additionally on combiner generators, the correlation cryptanalysis (Siegenthaler, 1985).

One well researched source of nonlinearity in ciphers is Boolean functions. In block ciphers, the nonlinearity often comes from Substitution Boxes or S-boxes which are actually a number of Boolean functions (hence also the name vectorial Boolean functions). On the other hand, in stream ciphers that nonlinearity comes from the Boolean functions. Both of those scenarios, while not the only ones, show us the important role of Boolean functions in cryptography. Finding Boolean functions fitting all the criteria and analyzing the best possible trade-offs between these criteria are still crucial questions today.

Historically, Boolean functions were predominantly used in conjunction with Linear Feedback Shift Registers (LFSRs). Two models that were most often used were filter generators and combiner generators. In a combiner generator, several LFSRs are used in parallel and their output is the input for a Boolean function. On the other hand, in a filter generator, the output is obtained by a nonlinear combination of a number of positions in one longer LFSR (see e.g. Carlet, 2010). Such Boolean functions need to be balanced, with high nonlinearity, large algebraic degree, large algebraic immunity, large fast algebraic immunity, and large correlation immunity (in the case of combiner generators).

We said that a Boolean function needs to be balanced (among other criteria) to be suitable for cryptography. Indeed this is true, but only when we consider the role of Boolean functions in filter and combiner generators. However, recently one more application emerged where we are actually interested in Boolean functions that have the minimal Hamming weight and are therefore as far as possible from being balanced. Such Boolean functions can be used to help resist side-channel attacks. These attacks do not rely on the security of the underlying algorithm, but rather on the implementation of the algorithm in a device (Mangard et al., 2007). One class of countermeasures against side-channel attacks are masking schemes. In masking schemes one randomizes the intermediate values that are processed by the cryptographic device. One obvious drawback of such an approach is the masking overhead which can be substantial in embedded devices or smart cards. Correlation immune Boolean functions can reduce the masking overhead either by applying leakage squeezing method (Carlet et al., 2012; Carlet and Guilley, 2014) or with Rotating S-box masking (Carlet and Guilley, 2013).

In order to obtain adequate Boolean functions, there exists a number of construction methods. Such methods can be roughly divided into algebraic constructions, random search, heuristics, and combinations of those methods (Picek et al., 2015c). In this paper, we examine one branch of heuristics, more precisely Evolutionary Algorithms (EAs) in order to evolve Boolean functions in accordance with the aforementioned properties. It is worth noting that EAs can be used either as the primary or the secondary construction method as it will be shown in the next section. In primary constructions one obtains new functions without using known ones. In secondary constructions, one uses already known Boolean functions to construct new ones (either with different properties or sizes) (Carlet, 2010). Furthermore, we experiment with Boolean function sizes that are of practical importance. However, such practical sizes also mean that the search space is very large. For a Boolean function with n inputs, there are in total 2^{2^n} possible Boolean functions.

We experiment with evolutionary algorithms that use three different represen-

tations: genetic algorithms (GAs) with binary encoding, genetic programming (GP) with tree representation and finally, Cartesian genetic programming (CGP) with graph representation. Besides the aforesaid single-objective algorithms, we experiment with a multi-objective approach. We emphasize that here we present only two possible applications of EAs for cryptography. There are many other possibilities, e.g. evolution of S-boxes (Picek et al., 2014b), finding search space parameters that can cause faults (Carpi et al., 2013), design of pseudo-random number generators (Lamenca-Martinez et al., 2006), and design of block ciphers (Hernandez-Castro et al., 2006), to name a few.

The remainder of this paper is organized as follows: Section 2 gives relevant information about Boolean functions, their representations and properties. In Section 3 we present a survey of related work. Next, in Section 4 we give our motivation for this research. In Section 4.1, we present the problem of finding minimal Hamming weight Boolean functions and their usage in masking schemes and then in Section 4.2 we discuss the necessary properties of a Boolean function to be used in combiner or filter generators. Section 5 gives our experimental setup, the algorithms we consider as well as the fitness functions for two design problems. Next, in Section 6, we give results for all algorithms and fitness functions as well as a short discussion and future work. Finally, in Section 7 we conclude the paper.

2 Preliminaries

Let n, m be positive integers, i.e. $n, m \in \mathbb{N}^+$. The set of all n-tuples of the elements in the field $\mathbb{F}_2$ is denoted as $\mathbb{F}_2^n$ where $\mathbb{F}_2$ is the Galois field with two elements. The inner product of two vectors $\vec{a}$ and $\vec{b}$ is denoted as $\vec{a} \cdot \vec{b}$ and equals $\vec{a} \cdot \vec{b} = \bigoplus_{i=0}^{n-1} a_i b_i$. Here, "$\oplus$" represents addition modulo two (bitwise XOR). The Hamming weight (HW) of a vector $\vec{a}$, where $\vec{a} \in \mathbb{F}_2^n$, is denoted $w_H(\vec{a})$. It is the number of non-zero positions in the vector, that is $w_H(\vec{a}) = \sum_{i=0}^{n-1} a_i$. The support $supp$ of a vector $\vec{a}$, where $\vec{a} \in \mathbb{F}_2^n$, is the set containing the non-zero positions in the vector $\vec{a}$.

An (n, m)-function is any mapping F from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. If m equals 1 then the function f is called a Boolean function.

A Boolean function f on $\mathbb{F}_2^n$ can be uniquely represented by a truth table (TT), which is a vector $(f(\vec{0}), ..., f(\vec{1}))$ that contains the function values of f, ordered lexicographically, i.e. $\vec{a} \leq \vec{b}$ (see e.g. Carlet, 2010).

The *support* ($supp$) of a Boolean function f is a set containing the non-zero positions in the truth table (TT) representation, i.e. $supp(f) = \{x : f(x) = 1\}$ (see e.g. Carlet, 2010). The HW of a Boolean function f is denoted by $w_H(f)$; it is the cardinality of its support.

The Walsh-Hadamard transform W_f is a second unique representation of a Boolean function that measures the correlation between $f(\vec{x})$ and the linear function $\vec{a} \cdot \vec{x}$ (see e.g. Carlet, 2010):

$$W_f(\vec{a}) = \sum_{\vec{x} \in \mathbb{F}_2^n} (-1)^{f(\vec{x}) \oplus \vec{a} \cdot \vec{x}}. \tag{1}$$

A third unique representation of a Boolean function f on $\mathbb{F}_2^n$ is a representation by means of a polynomial in $\mathbb{F}_2 [x_0, ..., x_{n-1}] / (x_0^2 - x_0, ..., x_{n-1}^2 - x_{n-1})$. This form is called the Algebraic Normal Form (ANF) (see e.g. Carlet, 2010). ANF is a multivariate

polynomial defined as (Meier et al., 2004):

$$f(\vec{x}) = \bigoplus_{\vec{a} \in \mathbb{F}_2^n} h(\vec{a}) \cdot \vec{x}^{\vec{a}}, \tag{2}$$

where $h(\vec{a})$ is defined by the Möbius inversion principle

$$h(\vec{a}) = \bigoplus_{\vec{x} \preceq \vec{a}} f(\vec{x}), \text{ for any } \vec{a} \in \mathbb{F}_2^n. \tag{3}$$

A Boolean function f is *balanced* if it takes the value 1 exactly the same number 2^{n-1} of times as value 0 when the input ranges over $\mathbb{F}_2^n$. For the Walsh-Hadamard transform, a Boolean function f is balanced if (Preneel et al., 1991):

$$W_f(\vec{0}) = 0. \tag{4}$$

The minimum Hamming distance between a Boolean function f and all affine functions (in the same number of variables as f) is called the *nonlinearity* of f. The nonlinearity Nl_f property of a Boolean function f can be expressed in terms of the Walsh-Hadamard coefficients as (Carlet, 2010):

$$Nl_f = 2^{n-1} - \frac{1}{2} \max_{\vec{a} \in \mathbb{F}_2^n} |W_f(\vec{a})|. \tag{5}$$

Function f has a *bias of nonlinearity* ϵ if it has the same output as its best affine approximation with probability of $\frac{1}{2} + \epsilon$ (Massey, 1995):

$$\epsilon = \frac{1}{2} - \frac{Nl_f}{2^n}. \tag{6}$$

A Boolean function f is *correlation immune* of order t (in brief, $CI(t)$) if the output of the function is statistically independent of the combination of any t of its inputs (Siegenthaler, 2006). For the Walsh-Hadamard spectrum it holds equivalently (Guo-Zhen and Massey, 1988):

$$W_f(\vec{a}) = 0, \text{ for } 1 \leq HW(\vec{a}) \leq t. \tag{7}$$

Boolean function f is *t-resilient* if it is balanced and with correlation immunity of degree t (Siegenthaler, 2006).

The *algebraic degree* deg of a Boolean function f is defined as the number of variables in the largest product term of the function's ANF having a non-zero coefficient (Carlet, 2010; MacWilliams and Sloane, 1977):

$$deg = \max(HW(\vec{a}) : h(\vec{a}) = 1). \tag{8}$$

Here, $h(\vec{a})$ is defined by the Möbius inversion principle.

The *algebraic immunity* (AI) of a Boolean function f is the lowest degree of the function g from $\mathbb{F}_2^n$ into $\mathbb{F}_2$ for which $fg = \vec{0}$ or $(f \oplus \vec{1})g = \vec{0}$ where f and g are Boolean functions. Here, fg is the Hadamard product of f and g, whose support is the intersection of the supports of f and g. A function g such that $fg = \vec{0}$ is called an annihilator of f (Carlet, 2010; Meier et al., 2004).

For a Boolean function f to resist fast algebraic attacks there should not be a Boolean function $g \neq \vec{0}$ of algebraic degree $1 \leq e \leq \lceil (\frac{n}{2}) \rceil$ and a Boolean function h

of a degree at most $n - e - 1$ such that $fg = h$ (Carlet and Tang, 2015). This property is defined as the *fast algebraic immunity* (FAI):

$$FAI = \min(2AI, \min\{deg(g) + deg(fg); 1 \leq deg(g) < AI\}). \quad (9)$$

For further information about Boolean functions, their properties and roles in cryptography, we refer interested reader to (Carlet, 2010; MacWilliams and Sloane, 1977).

3 Related Work

There exists a number of works that examine Boolean functions in cryptography and their generation with Evolutionary Computation (EC) techniques. Here, we try to enumerate them as exhaustively as possible.

As far as the authors know, the first application of genetic algorithms (GAs) to the evolution of cryptographically suitable Boolean functions is done by Millan et al. (1997) where the authors experiment with GA to evolve Boolean functions with high nonlinearity.

In his thesis, Clark presents several applications of optimization techniques in the field of cryptology (Clark, 1998). One of the applications is the evolution of Boolean functions with high nonlinearity using GA and hill climbing techniques.

Millan et al. (1998) use GA to evolve Boolean functions that have high nonlinearity. In conjunction with the GA they use hill climbing together with a resetting step in order to find Boolean functions with even higher nonlinearity and sizes of up to 12 inputs. They find balanced Boolean functions with eight inputs that have nonlinearity 112 and correlation immunity equal to one. Furthermore, when using GA with hill climbing they find eight input balanced Boolean functions with nonlinearity equal to 116. Millan et al. (1999) use variations of hill climbing method in order to find Boolean functions that have high nonlinearity and low autocorrelation.

Clark and Jacob (2000) experiment with two-stage optimization to generate Boolean functions. They use a combination of simulated annealing (SA) and hill climbing with a cost function motivated by Parseval theorem in order to find functions with high nonlinearity and low autocorrelation.

Clark et al. (2002) use simulated annealing to generate Boolean functions with cryptographically relevant properties. In their work, they consider balanced function with high nonlinearity and with the correlation immunity property less than or equal to two.

Kavut and Yücel (2003) develop improved cost functions for a search that combines SA and hill climbing. With that approach, the authors are able to find some functions of eight and nine inputs that have a combination of nonlinearity and autocorrelation values previously unattained. They also experiment with three-stage optimization method that combines SA and two hill climbing algorithms with different objectives.

Clark et al. (2003) experiment with SA in order to design Boolean functions by spectral inversion. They observe that many cryptographic properties of interest are defined in terms of Walsh-Hadamard transform values. Therefore, they work in the spectral domain where the cost function punishes those solutions that are not valid Boolean functions.

Burnett et al. (2004) present two heuristic methods where the goal of the first method is to generate balanced Boolean functions with high nonlinearity and low autocorrelation. The second method aims at generating resilient functions with high nonlinearity and algebraic degree that maximizes the Siegenthaler inequality.

Millan et al. (2004) propose a new adaptive strategy for local search algorithm for generation of Boolean functions with high nonlinearity. Additionally, they introduce the notion of the graph of affine equivalence classes of Boolean functions.

Burnett (2005) in her thesis uses three heuristic techniques to evolve Boolean functions. The first method aimed to evolve balanced functions with high nonlinearity. The second method is used to find balanced Boolean functions with high nonlinearity that are correlation immune. The last method is used to find balanced functions with high nonlinearity and propagation characteristics different from zero. Furthermore, she experiments with the evolution of S-boxes.

Aguirre et al. (2007) use a multi-objective random bit climber to search for balanced Boolean functions of size up to eight inputs that have high nonlinearity. Results indicate that the multi-objective approach is highly efficient when generating Boolean functions that have high nonlinearity.

Izbenko et al. (2008) use a modified hill climbing algorithm to transform bent functions to balanced Boolean functions with high nonlinearity.

McLaughlin and Clark (2013) on the other hand use SA to generate Boolean functions that have optimal values of algebraic immunity, fast algebraic resistance and algebraic degree. In their work, they experiment with Boolean functions of sizes up to 16 inputs.

Picek et al. (2013) experiment with GA and GP to find Boolean functions that possess several optimal properties. As far as the authors know, this is the first application of GP to the evolution of cryptographically suitable Boolean functions.

Hrbacek and Dvorak (2014) experiment with CGP to evolve bent Boolean functions of size up to 16 inputs. The authors investigate several configurations of algorithms in order to speed up the evolution process. Since they do not limit the number of generations, they succeed in finding bent function in each run for sizes between 6 and 16 inputs.

Several EAs are used by Picek et al. (2014a) to evolve Boolean functions that have better side-channel resistance. This paper presents the first application of optimization techniques to Boolean functions with improved side-channel resistance. With the goal of finding maximal nonlinearity values of Boolean functions Picek et al. (2014c) experiment with several EAs. Furthermore, they combine optimization techniques with algebraic constructions in order to improve the search. Although they are unable to find a balanced Boolean function with nonlinearity equal to 118, they present several possible avenues to follow when looking for highly nonlinear balanced Boolean functions. Picek et al. (2015c) compare the effectiveness of CGP and GP approach when looking for highly nonlinear balanced Boolean functions of eight inputs. Picek et al. (2015a) investigate several EAs 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 Picek et al. (2015b).

Mariot and Leporati (2015b) use Particle Swarm Optimization (PSO) to find Boolean functions with good trade-offs of cryptographic properties for dimensions up to 12. The same authors use GAs where the genotype consists of the Walsh-Hadamard values in order to evolve semibent (plateaued) Boolean functions (Mariot and Leporati, 2015a).

Picek et al. (2016) conduct a detailed analysis of the efficiency of a number of evo-

lutionary algorithms and fitness functions for Boolean functions with 8 inputs.

4 Motivation and Contributions

In this section, we present the reasoning behind the choice of the two scenarios explored in this paper as well as our contributions.

4.1 Boolean Functions and Masking

4.1.1 Masking with Codewords

Some applications manipulate sensitive data, such as cryptographic keys. Obviously, such data should remain secret. But skillful attackers might try to probe bits within a processor or a memory; they often succeed, unless countermeasures are implemented.

In a view to protect secrets from probing attempts, it is customary to implement a countermeasure known as masking. It consists in changing randomly the representation of the key (and of any other data which depends on the key), to deceive the attacker. For example, if each bit k_i, $1 \leq i \leq n$ of a key k is masked with a random bit m_i, then an attacker can only probe $k_i \oplus m_i$. However, provided m_i is uniformly distributed, the knowledge of $k_i \oplus m_i$ does not disclose any information on bit k_i, unless of course the attacker can also probe separately m_i, in which case a higher order masking would be necessary.

Now, for implementation reasons, it is often impractical to mask each bit individually. Indeed, the generation of unpredictable random numbers is costly, thus limiting the number of required random bits is to be welcomed. Furthermore, masking each bit of a vector of length n would correspond to choosing the global mask in the whole set of 2^n possible masks, which may be too costly. Specifically, on the example of the AES cipher, some key bytes are mixed with plaintext bytes and enter a Substitution Box (S-Box). Generating all the 256 S-Boxes suitable for all possible masks in $\mathbb{F}_2^8$ is expensive (especially for embedded systems). Hence, by restricting the number of possible masks, the overhead incurred by the countermeasure becomes more affordable.

Let us consider the simple example of masking one byte ($n = 8$). This can be achieved by using two complementary masks, such as $m_0 = (00000000)_2$ and $m_1 = (11111111)_2$. An attacker who measures one bit of the masked byte cannot derive any information about the corresponding unmasked bit. If we define by f the Boolean function $\mathbb{F}_2^8 \to \mathbb{F}_2$ whose support is $\{m_0, m_1\}$, then f plays its masking role as it is balanced: any bit can take value 0 and 1 with equal probability. In general, any Boolean function which is $CI(1)$ is a valid masking.

Let us now consider a stronger attacker who is able to probe two bits simultaneously. In this case, some information can be recovered. Typically, if the two masked bits are equal, then so are the two unmasked bits. So, by testing all pairs of bits, the attacker can recover the whole key (precisely: the whole key or its complement). It happens that, against such "second-order attacker", it would be desirable that the masks be the support of a $CI(2)$ Boolean function. But clearly, this support must have a cardinality strictly greater than 2.

Hence, masking can be summarized as the problem of finding Boolean functions whose support is the masks' set, with the two following constraints:

1. it should have small Hamming weight, for implementation reasons, and
2. it should have high correlation immunity t to resist an attacker with multiple ($\leq t$) probes.

Clearly, there is a tradeoff which motivates the research for low Hamming weight high correlation immunity Boolean functions.

4.1.2 Example of Masking suitable for the AES

An interesting example concerns the AES block cipher. It uses the same S-Box sixteen times in each cipher round. Thus fast implementations (for example in hardware) instantiate sixteen identical copies of the S-Box. If each S-Box is masked with a different mask and mapped randomly to each byte of the state, then, the masking has no overhead. The question is thus to find masks as codewords of a code of length $n = 8$, dimension $k = 4$, and maximum dual distance. The solution is reached by a $[8, 4, 4]$ auto-dual code (this is the indicator of an 8-input $CI(3)$ Boolean function of Hamming weight 16) (Carlet and Guilley, 2013).

4.1.3 State-of-the-art and Open Issues

Some work on the topic of *low Hamming weight high correlation immunity* Boolean functions has been summarized by Hedayat et al. (1999) in their book. However, in this book, some entries (minimum Hamming weight for a given pair (n, t)) are expressed as non-tight bounds. Recently, the exact value for entries corresponding to $(n = 9, t = 4)$, and $(n = 10, t \in \{4, 5\})$ have been obtained by Carlet and Guilley (2014, Table 2, page 66), using a computer exhaustive search with a Satisfiability Modulo Tool (SMT).

Let $w_{n,t}$ be the lowest weight of $CI(t)$ Boolean functions with n-bit inputs. Some remarkable values can be computed thanks to mathematical results:

- When $t = 1$, then the function must be nonzero and balanced, thus the Hamming weight of the function is greater or equal than 2, and the indicator of the parity-check code (that is, the set of even-weight words) (of parameters $[n, n-1, 2]$) reaches this bound. Hence $\forall n > 0, w_{n,1} = 2$.
- Now, it is well-known that the only nonzero Boolean function having this property is the constant function 1 (this can be easily shown by using for instance the inverse Fourier transform formula). Thus $w_{n,n} = 2^n$.
- A corollary of the Fon-De-Flaas theorem states that $w_{n,t} = 2^{n-1}$ for all $n > 0$ and $\lceil \frac{2n-2}{3} \rceil \leq t \leq n - 1$ (see Carlet and Guilley, 2014, Corollary 4.7, page 61).

But for $2 < t < \lceil \frac{2n-2}{3} \rceil$, there is no known mathematical procedure to compute $w_{n,t}$.

Thus, the method to find $w_{n,t}$ in those cases consists in (*i*) deriving mathematically some lower bounds, and (*ii*) finding some codes for which the lower bound is tight.

Regarding the requirement (*i*) of the method, examples of lower bounds are as follows (Carlet and Guilley, 2014, Lemma 4.2, page 59):

- for $n \geq 1$ and $d > 1$, $w_{n,t} \geq w_{n,t-1}$;
- for $n \geq 1$ and $d \geq 1$, $w_{n,t} \geq \frac{1}{2} w_{n+1,t}$;
- for $n > 1$ and $d > 1$, $w_{n,t} \geq 2 w_{n-1,t-1}$.

Now, condition (*ii*) can consist in selecting Boolean functions as indicators of binary codes. Indeed, a Boolean function is $CI(t)$ if and only if its support is a (not necessarily linear) code of dual distance strictly larger than t (see Carlet, 2010). One solution consists in looking into databases, such as in (Grassl, 2007). However, codes in this database are linear, whereas some functions of minimal Hamming weight are indicators of non-linear codes. This is the case for example for $8 \leq n \leq 11$ and $t = 2$

(for which $w_{n,t} = 12$) and for $9 \leq n \leq 12$ and $t = 3$ (for which $w_{n,t} = 24$). Obviously, the cardinality of support of a linear code is a power of two, which is not the case of 12 or 24.

Thus, it is required to find codes, possibly nonlinear, by unconventional methods. Indeed, as said, conventional constructions (Bose Ray-Chaudhuri Hocquenghem codes, Reed-Muller codes, Quadratic Residue codes, etc. (MacWilliams and Sloane, 1977)) favor linear codes, and furthermore do not have Hamming weight minimality as a constraint. Thus, Boolean Satisfiability Problem (SAT) and SMT solvers are an option, but fail for problems with more than a few thousand variables (i.e. when $n > 10$).

Now, it is well known that large S-Boxes make the design of block ciphers easier against cryptanalysis (Heys and Tavares, 1994). In addition, the fraction of good S-Boxes increases dramatically with the number of input variables (Youssef and Tavares, 1995). Even data bitwidths ($n \in 2\mathbb{N}^+$) are interesting from an implementation standpoint (for instance, Seberry et al. (1993) study a 12 bit input S-Box), whereas with odd data bitwidth ($n \in 2\mathbb{N} + 1$) it is easier to attain S-Box of high quality (Detombe and Tavares, 1992). In software, the memory has become widely available. For instance, a table with 2^{16} entries is now not a practical issue. In hardware, the integration capability has also drastically improved. Besides, generic power-efficient (Bertoni et al., 2004) and area-efficient (Canright, 2005) methods have been developed, which makes the area-overhead in hardware more acceptable. Thus, it is interesting to study values of n up to the reasonable value $n = 16$ (for which the S-Box will have 65536 entries, which is large, but perfectly manageable in multi-million gate circuits).

This is a motivation for the choice of the first experimental scenario. For instance, the exact values $(n, t) \in \{(11, 4-5), (12, 4-6), (13, 4-7)\}$ that are of practical interest, remained unknown until this research. To conclude, our **first** optimization problem is to evolve Boolean functions that have certain order of correlation immunity and minimal Hamming weight. Furthermore, we do this for Boolean function of sizes between 11 and 13 inputs in an effort to find the exact remaining unknown values, but also for sizes 14 to 16 inputs to additionally investigate whether EAs can handle larger sizes.

4.2 On Combiner and Filter Generators

Recall from Section 1 that symmetric key cryptography is most often divided into block and stream algorithms. In stream ciphers the encryption is made bitwise, through the addition, most often mod 2, of a *keystream* of the same size as the plaintext, output is by a pseudo-random generator (PRG) parameterized by a secret key. The resulting ciphertext can be decrypted by the same bitwise addition of the keystream, which gives back the plaintext. Stream ciphers are meant to be used on lighter devices than block ciphers and are supposed to be faster.

This is a difficult challenge for stream ciphers, since their security is dependent on the single choice of the PRG (they do not have the advantage of involving several rounds like block ciphers). In addition, nowadays the situation is still more difficult because modern block ciphers like the AES are very fast. We describe now briefly what are the criteria for the design of a stream cipher; more can be found in (Carlet, 2013). For reasons of efficiency, the PRG contains a linear part, initialized by means of a secret key and possibly a binary string sent more often over a public channel, called an initial vector, or initial value (IV). The keystream is made of the bits output by the PRG during a sufficient number of clock cycles, the linear part being updated at each clock cycle by a bijective linear function, and its content being nonlinearly combined in some way to produce the output bit. As observed already by Claude Shannon in the forties (Shan-

non, 1949), every cipher can be translated into a system of equations whose coefficients are deduced from the public data and from the observed data (the attacker being supposed to have access to a part of the keystream and needing to recover the rest of it); the unknowns of the equations can be the secret key bits or (in the case of stream ciphers) the initialization of the PRG. Note that the description of the PRG itself is supposed to be public, only its initialization being kept secret, since it is usually believed that making a part of the description secret, which requires the use of a part of the secret key for this description, reduces too much the possible size of the PRG and gives less robust systems than using the whole key-IV for generating the initialization. These equations need to have many unknowns and to be nonlinear for ensuring resistance to the possible cryptanalyses using this method. This nonlinearity is often ensured thanks to a *Boolean function* which, in the so-called *filter model*, takes as input n bits chosen at fixed positions in the linear part of the PRG at its current state (n being much smaller than the size of the linear part, for reasons of speed). In the simplest version of PRG, the function itself outputs the current bit of the keystream and the choice of the function must ensure resistance to all known attacks (in practice, additional devices such as memory cells are added, but this simplest model of PRG must have been proven already resistant to all known attacks). If a function is weak against one of the existing attacks, it does not have practical interest, even if it is very strong against all the other attacks. Note that mainstream products in the industry are based on other stream ciphers (e.g. the eStream finalists). However, in the governmental market, in very light weight cryptography, or in "internal cryptography" dedicated to the inner protection of a chip, the filter model is often preferred, because its design has been extensively analyzed and has a low gate count.

4.2.1 Algebraic Attacks on Stream Ciphers

Denoting by $(u_1, \cdots, u_N)$ the initialization of the linear part of the PRG, by L its bijective update linear function, by L' the (linear) projection giving from the current state of the linear part the bits used as input to f, and by s_i the i-th bit of the keystream, the equations mentioned by Claude Shannon (see above) have the form:

$$s_i = f(L' \circ L^i(u_1, \cdots, u_N)), \tag{10}$$

where $L' \circ L^i$ represents the composition of functions L' and L^i. Since $L' \circ L^i(u_1, \cdots, u_N)$ is linear and then has degree 1 in $u_1, \cdots, u_N$, all the equations have the same global degree, equal to the algebraic degree deg of f.

Courtois and Meier (2003) improved Shannon's attack by observing that, if a nonzero n-variable Boolean function g and a Boolean function h, both of low algebraic degrees, can be found such that $h = fg$, this allows multiplying each term of Eq. (10) by $g(L' \circ L^i(u_1, \cdots, u_N))$ to obtain the equation:

$$s_i\, g(L' \circ L^i(u_1, \cdots, u_N)) = h(L' \circ L^i(u_1, \cdots, u_N)), \tag{11}$$

which has an algebraic degree of h if $s_i = 0$ and an algebraic degree of $g + h$ if $s_i = 1$. Since both g and h have low algebraic degrees, the system can then be solved.

As observed by Meier et al. (2004), since $fg = h$ implies $fh = f^2g = fg$, that is, $f(g + h) = \vec{0}$ (recall that the addition of such Boolean functions is mod 2), the existence of $g \neq \vec{0}$ and h of algebraic degrees at most d, such that $fg = h$, is equivalent to the existence of $g \neq \vec{0}$ of algebraic degree at most d such that $fg = \vec{0}$ (that is, g is an annihilator of f) or $(f + \vec{1})g = \vec{0}$ (that is, g is an annihilator of $f + \vec{1}$, this case happens when $g = h$ in the original equality $fg = h$).

Other algebraic-like attacks exist.
- The Rønjom-Helleseth attack (Rønjom and Helleseth, 2007) is efficient if the algebraic degree of the function is not close to n and has complexity $O(N^{deg(f)})$.
- The *Fast Algebraic Attack* (FAA), introduced by Courtois (2003) shortly after the standard algebraic attack is efficient if one can find g of low algebraic degree and $h \neq \vec{0}$ of algebraic degree significantly less than n (but maybe larger than $\lceil \frac{n}{2} \rceil$) such that $fg = h$. Its complexity is roughly of the order (see Hawkes and Rose, 2004):

$$O\left(\min\left\{N^{\max[\deg(g)+\deg(fg),3\deg(g)]}, g \neq 0\right\}\right).$$

This, and the facts that the FAA with $g = \vec{1}$ is less efficient than the Rønjom-Helleseth attack and that the FAA with $deg(g) \geq AI(f)$ results in the algebraic attack, has led in (Carlet and Tang, 2015) to the study of so-called *Fast Algebraic Complexity*:

$$FAC(f) := \min\left\{\max\left[\deg(g) + \deg(fg), 3\deg(g)\right]; 1 \leq \deg(g) < AI(f)\right\},$$

whose value is invariant by changing f into $f + \vec{1}$, is bounded above by n, and is bounded below by the Fast Algebraic Immunity (Carlet, 2013; Carlet and Tang, 2015).

Let an n-variable Boolean function f, with (optimal) algebraic immunity $\lceil n/2 \rceil$. We assume it is used either as a combiner or as a filter in a stream cipher, and that the linear part of the pseudo-random generator (from which the Boolean function takes its n input bits to produce a bit of the keystream at each clock-cycle) has size N. Then the complexity of an algebraic attack using one annihilator of degree $\lceil n/2 \rceil$ is roughly $7\left(\binom{N}{0} + \cdots + \binom{N}{\lceil n/2 \rceil}\right)^{\log_2 7} \approx 7\left(\binom{N}{0} + \cdots + \binom{N}{\lceil n/2 \rceil}\right)^{2.8}$ (Courtois and Meier, 2003). Let us choose $N = 256$ (which is usual), then the complexity of the algebraic attack is at least 2^{80} (which is considered nowadays a sufficient complexity) for $n \geq 13$.

4.2.2 Constraints and Bounds on Boolean functions for Stream Ciphers

1. The function must be *balanced*, since otherwise, the attacker would be able to distinguish a randomly chosen pair, from a pair "plaintext-ciphertext" (or even a part of the plaintext and a part of the ciphertext, both at the same positions) by calculating the Hamming distance between the two texts of the pair.

2. The function must have large *algebraic degree* to allow resistance to the Berlekamp-Massey attack (Massey, 1969) and to the already mentioned Rønjom-Helleseth attack (Rønjom and Helleseth, 2007). In practice, we want an algebraic degree close to $n - 1$ (the maximum for a balanced function).

3. The function needs also to allow resistance to the fast correlation attack (Meier and Staffelbach, 1988). This attack uses the existence of an affine Boolean function g (that is, a function of algebraic degree at most 1) whose Hamming distance to f is small, i.e. to withstand this attack, the *nonlinearity* should be high. A large nonlinearity Nl_f is analogous to the small bias of nonlinearity $\epsilon = \frac{1}{2} - \frac{Nl_f}{2^n}$ (the smallest possible bias is $2^{-n/2-1}$, with bent functions; but bent functions are unbalanced, so we must accept larger bias; say ϵ approximately equal to $2^{-n/2}$ or even $2^{-n/2+1}$).

   The nonlinearity of an n-variable Boolean function is bounded above by $2^{n-1} - 2^{n/2-1}$. The functions whose nonlinearity equals this maximal value

$2^{n-1} - 2^{n/2-1}$ are *bent*. A function f used in a stream cipher should not be bent because bent functions are never balanced, but it should have nonlinearity near this maximum, since the fast correlation attack has an on-line complexity proportional to $\left(\frac{1}{\epsilon}\right)^2$. Note that the average nonlinearity of random n-variable functions lies asymptotically near $2^{n-1} - 2^{n/2-1}\sqrt{2n \ln 2}$ (see Rodier, 2006), which is not bad.

4. The function must have *algebraic immunity* close to $\lceil n/2 \rceil$, and fast algebraic complexity near n; for this, having a *fast algebraic immunity* FAI near n is sufficient. A difference by 1 in the algebraic immunity can have a large impact on the complexity of the algebraic attack, (see Canteaut, 2006; Carlet, 2010).

5. In some applications it is important to *minimize the number of terms in the ANF* because minimizing the number of terms in the ANF allows faster computing the output of the function.

6. A particular model of PRG is the combiner model, in which the n input bits to the Boolean function are the outputs to n independent Linear Feedback Shift Registers (LFSRs). A divide-and-conquer attack called *correlation attack* obliges the function to be k-resilient for a sufficiently large value of k (see Camion et al., 1992; Siegenthaler, 1985), that is, satisfy $W_f(a) = 0$ for every vector a of Hamming weight at most k. No class of such resilient functions having optimal algebraic immunity has been found yet. Moreover, the Siegenthaler bound shows that resilient functions cannot resist the Rønjom-Helleseth attack or the FAA; hence, the combiner model is not widely used nowadays and the filter model (where the linear part is in general made of a single LFSR generating a sequence of maximal period) is preferred. For a resilient Boolean function where $t > 1$ and $t \neq n-1$, the following Siegenthaler bound holds (Siegenthaler, 2006):

$$t \leq n - deg - 1. \tag{12}$$

Therefore, the correlation immunity and the algebraic degree properties are conflicting and it is not possible to obtain a Boolean function with both properties optimal.

The goal in our **second** experimental set is to evolve Boolean functions that possess the aforesaid properties and have dimensions of practical interest, i.e. 13 inputs and higher.

5 Experimental Setup

In this section, we briefly present the algorithms we use as well as the experimental setup and fitness functions for our research scenarios. For all the experiments, we use Evolutionary Computation Framework (ECF) (Jakobovic, 2014). We note that ECF has all the algorithms and fitness functions we define here.

5.1 Genetic Algorithm

The GA represents the individuals as strings of bits which present truth tables of Boolean functions. We use a 3-tournament selection, where the worst from the 3 randomly selected individuals is eliminated (Eiben and Smith, 2003). A new individual

is created by applying crossover to the remaining two and then a mutation with given probability (Algorithm 1).

Mutation is selected uniformly at random between a simple mutation, where a single bit is inverted, and a mixed mutation, which randomly shuffles the bits in a randomly selected subset. When the balancedness property is included in the fitness function, we include a balanced simple mutation; in this operator two bits are randomly inverted if the solution is balanced, and only one bit otherwise. The crossover operators are one-point and uniform crossover, performed uniformly at random for each new offspring. For each of the objectives we experiment with population sizes of 50, 100, 500, and 1 000 and individual mutation probabilities of 0.1, 0.3, 0.5, 0.7, and 0.9. The mutation probability is used to select whether an individual would be mutated or not, and the mutation operator is executed only once on a given individual. For example, if the mutation probability is 0.7, then on average 7 out of every 10 new individuals will be mutated and one mutation will be performed on each of those individuals.

Algorithm 1 Steady-state tournament selection

randomly select k individuals;
remove the worst of k individuals;
$child$ = crossover (best two of the tournament);
perform mutation on $child$, with given individual mutation probability;
insert $child$ into population;

5.2 Genetic Programming

Genetic programming (GP) uses a representation where individuals are trees of Boolean primitives which are then evaluated according to the truth table they produce. The function set for GP in all experiments is OR, XOR, AND, XNOR, and AND with one input inverted. Terminals correspond to n Boolean variables. GP uses the same selection as in Algorithm 1 with tournament size 3. The crossover is performed with five different tree-based crossover operators selected at random: a simple tree crossover with 90% bias for functional nodes, uniform crossover, size fair, one-point, and context preserving crossover (Poli et al., 2008). We use a single mutation type, a subtree mutation applied with 30% probability, and experiment with maximum tree depth sizes of 5, 7, 8, and 9 and population sizes of 200, 500, 1 000, and 2 000.

5.3 Cartesian Genetic Programming

Cartesian genetic programming (CGP) (Miller, 1999) solutions are directed graphs with Boolean primitives as nodes, that are also evaluated using the truth table they produce. The function set for the CGP is the same as for the GP. The number of input connections n_n for each node is two and the number of program output connections n_o is one. Setting the number of rows to be 1 and levels-back parameter to be equal to the number of columns is regarded as the best and most general choice (Miller, 2011). We experiment with genotype sizes of 500, 1 000, 2 000, and 3 000 nodes and mutation rates of 1%, 4%, 7%, 10%, and 13% per node. For CGP individual selection we use a (1 + 4)-ES in which the offspring are favored over parents when they have a fitness better than or equal to the fitness of the parent. The population size for CGP equals five in all our experiments. The mutation operator is one-point mutation where the mutation point is chosen with a fixed mutation rate. The number of genes mutated is defined as fixed percentage of the total number of genes. The single output gene is never mutated and is taken from the last node in the genotype. A gene chosen for mutation might be a node representing an

input connection or a function.

5.4 Common Parameters

The number of independent runs for each experiment is 30. For the stopping condition we use the number of evaluations, which we set to 1 000 000.

5.5 Fitness Functions

5.5.1 The First Design Problem

The first problem (Section 4.1) concerns evolving Boolean functions with a given target order of correlation immunity and minimal Hamming weight. The fitness function is designed as follows, where the goal is **maximization**:

$$fitness_1 = (MAX\_HW - supp) - MAX\_HW \times |CI - TARGET\_CI|. \quad (13)$$

Here, MAX_HW represents the Hamming weight of a Boolean function that has all ones in its truth table (i.e. $HW = 2^n$, where n represents the number of inputs of a Boolean function), $TARGET\_CI$ is the order of the correlation immunity we want to obtain and finally, $supp$ represents the cardinality of the support of a Boolean function. The function consists of two parts: the first part rewards Boolean functions with smaller support, while the second part acts as a penalty for solutions with different target correlation immunity. The penalty part is multiplied with maximum value of the reward part, so that any solution with the right CI is always better than any other solution with CI different that the target value. This way, the distance to the target CI is regarded as the primary, and the support as the secondary objective.

Furthermore, we use a condition that eliminates trivial solutions, i.e. Boolean functions with Hamming weight equal to either zero or MAX_HW. If such a solution is encountered, it is given the worst fitness value, which is $-MAX\_HW \times n$.

Multi-objective Optimization

Since our first scenario of the function design includes two criteria, this problem can also be formulated as multi-objective optimization. The first criterion is attaining a desired target correlation immunity, and the second one is the maximization of the support. Following these criteria, a multi-objective problem can be formulated as:

$$objective_A = |CI - TARGET\_CI|; \quad (14)$$

$$objective_B = MAX\_HW - supp, \quad (15)$$

where the first criterion, $objective_A$, is **minimized**, while the second criterion, $objective_B$, is **maximized**.

In our experiments we apply the well known NSGA-II algorithm for multi-objective optimization (Deb et al., 2002). Note that NSGA-II can be paired with any of the Boolean representations (i.e. truth table in GA, tree in GP, and graph in CGP), but based on the performance in the initial round of experiments, we only present the results of the tree representation (GP) with the multi-objective evolution.

5.5.2 The Second Design Problem

The second design problem is concerned with evolving Boolean functions with properties as described in Section 4.2.2. In this scenario we aim to optimize multiple properties that can oppose each other; therefore we decide to apply a bottom up approach where

we begin with a simple fitness function including only a single property, and then optimize more complex fitness functions that include additional properties of interest. We follow this approach since our previous results indicate that concentrating on a fitness function which includes all the properties can actually result in a solution with poor property values (Picek et al., 2013).

Before presenting fitness functions for the second design problem, we give a short discussion on two properties of interest: algebraic immunity and fast algebraic immunity. To evaluate a Boolean function with regards to those two properties is exponentially dependent on the number of inputs in computation time. Each of these properties for input dimension greater than 12 require several orders more time to calculate them (for instance, to calculate fast algebraic immunity of a Boolean function with 14 inputs we need more than 5 minutes). Therefore, it is not practicable to include these two properties in our fitness functions since we work with 1 000 000 evaluations and 30 independent runs. Because of that, we calculate those two properties a posteriori and present the results. We note that this also presents one serious drawback of heuristics when evolving such Boolean functions, since it is unrealistic to run complex calculations for a great number of evaluations. Still, we show that evolving Boolean functions with regards to other properties can result in functions that are also relatively fit in regards to these two properties.

For every fitness function, the goal is *maximization*. We start our experiments with the simplest fitness function that consists of one constraint (the function needs to be balanced) and the nonlinearity property. Here we use a two stage fitness in which a fitness bonus equal to the nonlinearity is awarded only to a genotype that is perfectly balanced (this occurs when $BAL = 0$); otherwise, the fitness is only the balancedness penalty. The balancedness penalty BAL is defined as the negative difference up to the balancedness (i.e. the number of bits that need to be changed to reach balancedness). The delta function $\delta_{BAL,0}$ takes the value one when $BAL = 0$ and is zero otherwise.

$$fitness_1 = BAL + \delta_{BAL,0} Nl_f. \tag{16}$$

The second fitness adds the algebraic degree property. Here, the fitness equals the sum of the nonlinearity and algebraic degree properties when the function is balanced, and the balancedness penalty otherwise.

$$fitness_2 = BAL + \delta_{BAL,0}(Nl_f + deg). \tag{17}$$

Next, we add the minimization of the number of terms in ANF representation ($ANFMinimize$). This is the first time, as far as the authors know, that this criterion is considered in the evolution of Boolean functions. Therefore, we do not have a priori intuition on the obtainable solutions. In accordance with this, we set the $ANFMinimize$ criterion to be of secondary importance:

$$fitness_3 = BAL + \delta_{BAL,0}(Nl_f + deg + ANFMinimize). \tag{18}$$

Here, we want to achieve the smallest possible number of terms in the ANF representation and since the problem is presented as the *maximization*, we calculate it as:

$$ANFMinimize = (2^n - HW(ANF))/HW(ANF). \tag{19}$$

In this way, the reward for lowering the weight of the ANF representation is in range $[0, 1]$. When compared to the values that other properties can achieve, we see that

this property influences the evolutionary process only marginally (since it is considered of secondary importance in our setup). Finally, for situations where we are interested in combiner generators, we add the correlation immunity property ($fitness_4$). Our previous results show that if both correlation immunity and algebraic degree properties are present in the fitness function, algebraic degree value dominates over the correlation immunity (Picek et al., 2013). However, we still allow that these two properties freely compete for higher values. If one needs to obtain a Boolean function with a specific order of correlation immunity, the fitness is easily adaptable to version where algebraic degree is bounded in accordance with the Siegenthaler inequality.

$$fitness_4 = BAL + \delta_{BAL,0}(Nl_f + deg + ANFMinimize + CI). \tag{20}$$

We note that to calculate the nonlinearity property, it is also possible to use the spectrum based cost function (Clark et al., 2004). However, we opted not to follow that line of work for the following reasons: the first one is that good values for additional parameters X and R for Boolean functions of sizes 13 up to 16 inputs are not well researched. The second reason lies in the fact that our experiments show that it is possible to evolve highly nonlinear Boolean functions even with the simplest calculation of the nonlinearity part of the fitness function.

Multi-Objective Optimization

The multi-objective approach can also be taken in the second scenario. In this case, we first have to identify the objectives. First of all, the balancedness property cannot be used as an independent criteria, since it is a constraint that needs to be satisfied and should therefore be incorporated in every objective. Secondly, the minimization of the number of terms in the ANF representation is considered as a secondary criterion, relevant only when all the other criteria are equal. Following this, we design the multi-objective approach taking the nonlinearity, algebraic degree, and correlation immunity into account. The multi-objective problem which considers those properties is therefore defined as:

$$objective_A = BAL + \delta_{BAL,0}Nl_f, \tag{21}$$
$$objective_B = BAL + \delta_{BAL,0}deg, \tag{22}$$
$$objective_C = BAL + \delta_{BAL,0}CI, \tag{23}$$

where all the objectives are maximized. If the solution is not balanced, the objective values adopt a negative unbalancedness penalty, so not to dominate over any criteria. Only if the solution is balanced, all the criteria assume their respective values.

6 Results and Discussion

In this section, we present the results for each of the algorithms used as well as all fitness functions. The first part deals with the results for the evolution of minimal Hamming weight Boolean functions with different orders of correlation immunity. The second part presents Boolean functions for usage in combiner and filter generators. We report the performance of the selected algorithms, and additionally present the best obtained values in order to compare EAs with the existing results. The evolutionary algorithms are first compared with each other using basic statistical indicators to assess their performance. After that, we only select the single best results obtained by any of the algorithms and compare it with the values found in the related literature.

6.1 Tuning Phase

There is a large number of experiments; more precisely, for the first design problem, we have 81 combinations for each of the target CI values and function sizes (11-16 inputs) and 16 combinations for the second design problem (13-16 inputs, 4 fitness functions). In accordance with that, we conduct a parameter tuning phase based on the first problem for a medium sized Boolean function of 14 inputs and the target correlation immunity order of four. Parameter tuning phase has a stopping condition of 1 000 000 evaluations. Afterward, we use the best obtained set of parameters for all test scenarios.

6.2 The First Design Problem

Here, we give results obtained for all algorithms and fitness function as in Eq. (13). We display for each algorithm the best value it obtained for a certain input size and the correlation immunity level as well as the number of times it reaches that value. We do not give further statistics in the form of average values or similar since we do not consider it descriptive. As an example, consider a set of very good solutions and one extremely bad solution. Although that algorithm performs very well, the penalty given to that bad solution will skew the complete results. Furthermore, in this application domain the user is typically interested only in the best obtained solution.

6.2.1 Genetic Algorithm

The results for GA in this application were very poor; in the tuning phase we were unable to obtain a single solution with the desired correlation immunity for any of the parameter settings. This suggests that bitstring GA with common genetic operators we used cannot traverse the search space towards better solutions. The GA does succeed in finding the desired values, but only for very small problem sizes (e.g. for up to six variables), where the size of the solution is not large. However, since those cases are not representative to the problem, we do not experiment with GA in the rest of the paper when considering the first design problem.

6.2.2 Genetic Programming Results

In the tuning phase the results indicate the best maximum depth is 5; for different population sizes for GP there are practically no statistical differences, so we opted for the size of 1 000. Depending on the number of bits and the target CI, the problem at hand may be so easy that every combination of parameters reaches the same solution every time, or so hard that there is a very small probability of reaching the best (optimal) fitness value regardless of the parameters.

The results for the GP for all combinations of sizes and target CI are given in Table 1. Because of the previous remark, the results are given in the form of the best obtained value (i.e. the minimal Hamming weight) and the number of runs (out of 30) in which that value was reached.

6.2.3 Cartesian Genetic Programming Results

When using CGP, we observed that the parameter values play much more significant role than in the GP case. For Boolean functions up to size of 13 inputs, the genotype size of 3 000 and mutation probability of 13% performs the best, but for larger sizes the best results are obtained for genotype size of 1 000 and mutation rate of 10%. In Table 2, we give the statistics for CGP in the form of the best obtained solutions and the number of times those solutions were reached. When the algorithm is not able to find a correct solution (by correct, we mean with a desired order of CI) we denote it with "-".

Table 1: GP results (best value/number of best runs).

t \ n	11	12	13	14	15	16
2	16/25	16/19	32/14	64/20	128/16	256/23
3	32/29	32/17	32/2	64/15	128/12	256/11
4	128/11	256/30	256/12	256/1	2048/14	4096/12
5	256/13	256/5	512/7	1024/16	2048/23	4096/23
6	512/13	1024/27	1024/5	2048/5	4096/3	4096/4
7	1024/16	1024/8	2048/4	4096/8	8192/18	8192/1
8	1024/30	2048/30	4096/30	8192/30	8192/1	16384/4
9	1024/30	2048/30	4096/30	8192/30	16384/30	16384/2
10	1024/30	2048/30	4096/30	8192/30	16384/30	32768/30
11		2048/30	4096/30	8192/30	16384/30	32768/30
12			4096/30	8192/30	16384/30	32768/30
13				8192/30	16384/30	32768/30
14					16384/30	32768/30
15						32768/30

Table 2: CGP results (best value/number of best runs).

t \ n	11	12	13	14	15	16
2	16/4	32/7	32/8	64/3	128/4	256/2
3	32/1	64/2	128/2	256/2	2048/8	1024/1
4	256/2	512/3	512/1	2048/3	4096/2	8192/1
5	512/2	1024/4	2048/7	4096/3	8192/1	8192/1
6	1024/25	2048/26	4096/28	8192/27	16384/17	32768/8
7	1024/23	2048/24	4096/27	8192/23	16384/8	32768/3
8	1024/22	2048/20	4096/21	8192/20	16384/4	-
9	1024/14	2048/12	4096/15	8192/8	16384/1	-
10	1024/5	2048/8	4096/9	8192/2	-	-
11		2048/1	4096/5	-	-	-
12			4096/1	-	-	-
13				-	-	-
14					-	-
15						-

6.2.4 Multi-objective Optimization Results

In this problem, the multi-objective (MO) approach did not prove competitive; the results for medium-sized problem with 13 bits are given in Table 3. Table entries with a dash denote that the algorithm did not reach the target CI value in this case. Since the MO results were significantly worse, we do not include them in further comparisons.

Table 3: MOGP results (best value).

n \ t	2	3	4	5	6	7	8	9	10	11	12
13	2048	4096	4096	-	-	-	-	-	-	-	-

6.2.5 Best Obtained Solutions

In this section, we compile the list of the best obtained solutions over all algorithms; the results are given in Table 4. The values given in bold present the ones that were previously unknown. Regarding the individual algorithms, if a solution is found only with GP, we leave the background color of a cell white. The cells with gray background, on the other hand, denote solutions found both with GP and CGP.

6.3 The Second Design Problem

In this section, we first present the results only as the obtained values of the related fitness functions, to compare different algorithms. For each combination of input size

Table 4: Best obtained results, the first design problem.

t \ n	11	12	13	14	15	16
2	16	16	32	64	128	256
3	32	32	32	64	128	256
4	**128**	256	256	256	2 048	4 096
5	**256**	**256**	512	1 024	2 048	4 096
6	512	**1 024**	1 024	2 048	4 096	4 096
7	1 024	1 024	**2 048**	4 096	8 192	8 192
8	1 024	2 048	4 096	8 192	8 192	16 384
9	1 024	2 048	4 096	8 192	16 384	16 384
10	1 024	2 048	4 096	8 192	16 384	32 768
11		2 048	4 096	8 192	16 384	32 768
12			4 096	8 192	16 384	32 768
13				8 192	16 384	32 768
14					16 384	32 768
15						32 768

Table 5: Optimal values of relevant cryptographic properties.

n	$Nl_f(bias)$	AI	deg	FAI
13	4 064 (0.0039)	7	12	12
14	8 128 (0.0039)	7	13	13
15	16 320 (0.0019)	8	14	14
16	32 640 (0.0019)	8	15	15

and fitness function, we give the best obtained value and the average value. Afterward, we single out only the best solutions for each fitness and input size and show the values of the relevant cryptographic properties. Table 5 shows the theoretical optimal values (maximal) of the observed properties. We omit the correlation immunity property from the table since its value is conflicting with the algebraic degree property and equals 0 when algebraic degree reaches the optimal value. Furthermore, we do not give data for the minimal number of terms in the ANF representation since that data is currently unknown.

6.3.1 Genetic Algorithm Results

For this problem we initially applied the GA only for Boolean functions with 13 inputs. The results are shown in Table 6 where for each fitness function we give the best and the mean obtained result from 30 runs. The GA is significantly worse than the other methods; this is not surprising, since already for 13 bits the size of the truth table is 8 192. Since the difference in the performance only increases for larger number of inputs, we omit the other GA results in this section.

Table 6: GA results (best/mean).

n \ fitness	1	2	3	4
13	3 964 / 3 958.2	3 974 / 3 968.2	3 984.5 / 3 976.9	3 980.5 / 3 976.7

6.3.2 Genetic Programming Results

Table 7 shows the best and mean fitness values for all the combinations of the number of bits and the defined fitness functions. It can be shown that all the results are significantly better than the GA results (the analysis is not included). One possible downside is that the evaluation takes considerably longer than the GA, since for each GP individ-

ual a complete truth table must be constructed to evaluate the relevant properties.

Table 7: GP results (best/mean).

n \ fitness	1	2	3	4
13	4 032 / 4 032	4 042 / 4 039.1	4 040 / 4 039.9	4 043 / 4 039.9
14	8 064 / 8 064	8 074 / 8 072.7	8 078 / 8 073.9	8 078 / 8 074.5
15	16 256 / 16 256	16 264 / 16 263.8	16 265 / 16 264.9	16 265 / 16 264.5
16	32 512 / 32 512	32 615 / 32 523.875	32 616 / 32 524.1	32 526 / 32 523.2

6.3.3 Cartesian Genetic Programming Results

We give the best and mean value for CGP algorithm in Table 8. It can be seen there are virtually no differences between fitness functions 2, 3, and 4. This indicates that the significance of the ANFMinimize and CI properties in the fitness function is not enough to drive the search. To rectify this, one can add the weight factor on the ANFMinimize and the correlation immunity parts or restrict algebraic degree only to a certain level (and thus allowing the CI property to assume higher values).

Table 8: CGP results (best/mean).

n \ fitness	1	2	3	4
13	4 032 / 3 981.6	4 042 / 3 988.8	4 042 / 3 988.8	4 042 / 3 991.9
14	8 064 / 7 413.2	8 075 / 7 431.1	8 075 / 7 432.7	8 075 / 7 420.5
15	16 128 / 15 346.4	16 140 / 15 351.9	16 140 / 15 351.9	16 138 / 15 364.1
16	32 256 / 31 636.6	32 268 / 31 587.45	32 268 / 31 587.5	32 264 / 31 602.4

6.3.4 Best Obtained Solutions

Finally, we combine all the single-objective optimization results and select only the best obtained solutions for every combination of input size and fitness function. For each selected solution, we present the observed cryptographic properties in Table 9. Apart from the number of terms in ANF representation (#ANF), all the properties are maximized. We do not explicitly write which algorithm obtained a certain solution since it is not always easy to select a single best solution. For instance, when $n = 13$ and fitness function is as in Eq. (18), the best GP result is given in table below. However, CGP finds a solution that has the nonlinearity equal to 4 030, algebraic degree equal to 12 and $\#ANF$ equal to 61. They both have the same fitness value, but the choice which Boolean function is better is dependent on the setting one considers.

Clearly, the single-objective fitness gives insufficient weight to the CI property to be able to reach higher values. This can be remedied with the multi-objective optimization, as shown below.

6.3.5 Multi-objective Optimization Results

In order to compare the attained property values, we present the multi-objective results after the best results from single-objective optimization. Table 10 shows the best non-dominated results from multi-objective optimization. The bold values in each row represent the property whose value was the best obtained, with other properties selected as large as possible.

In this problem the MO approach did succeed in finding larger values for certain properties, such as correlation immunity. The best values for nonlinearity and algebraic degree are about the same as in the single-objective case. If we are aiming for a more

Table 9: Best obtained results, the second design problem, Eq. (20).

n	fitness	Nl_f	deg	$\#ANF$	CI	AI	FAI
13	1	4 032	5	28	0	4	5
	2	4 032	10	55	0	3	4
	3	4 032	7	8	0	3	4
14	1	8 064	10	50	0	3	4
	2	8 064	13	11	0	3	4
	3	8 064	12	14	1	3	4
15	1	16 256	5	33	0	3	4
	2	16 256	8	25	0	3	4
	3	16 256	8	9	0	3	4
16	1	32 512	12	99	0	4	6
	2	32 512	14	12	0	3	4
	3	32 512	12	21	1	4	6

general all-round function with majority of properties being 'good enough', the single-objective is still the best option. However, if we need to maximize a single property, then the MO approach gives a viable alternative.

Table 10: MOGP results (non-dominated solutions with max. property values).

n	Nl_f	deg	CI
13	**4 032**	7	0
13	3 966	**12**	0
13	0	1	**11**
14	**8 064**	7	1
14	7 806	**13**	0
14	4 096	2	**11**
15	**16 128**	4	1
15	15 728	**13**	0
15	0	1	**11**
16	**32 512**	7	0
16	30 660	**14**	1
16	0	1	**12**

6.4 Discussion and Future Work

When considering the first design problem and the performance of CGP, we observe that the results are somewhat worse than those from related work (Picek et al., 2015a,c). Actually, CGP often outperforms the GP on Boolean problems and one can ask why is this not the case here. Naturally, it is also hard to expect that CGP will always outperform GP because there is a high influence of the stopping condition and fitness function. We observed in our experiments that CGP would benefit from higher number of evaluations in this problem, but to keep the comparison as fair as possible we retained the same number of evaluations as GP.

Furthermore, our experiments showed that it is possible to use a better fitness function for this problem when using CGP. Better CGP results can be obtained with fitness function with the goal of *minimization*:

$$fitness = -|TARGET\_SUPPORT - supp| - |TARGET\_CI - CI|. \quad (24)$$

One can see that this function uses additional info and that is the target level of support ($TARGET\_CI$). Naturally, if one knows what level one wants to achieve, this does not pose a problem, but in the case where the best support value (and that is the value one is usually interested in) is unknown, this represents a serious drawback. Of course, it is possible to iteratively decrease the level of support until the algorithm cannot find such a value anymore, but that substantially prolongs the evolution process.

However, we note that when using this fitness function, CGP easily outperforms GP. A plausible scenario for this fitness function could be to first use fitness as in Eq. (13) with GP to find out what is the best support value, and then to use fitness as in Eq. (24) and CGP to find as many as possible Boolean functions that have desired support value.

When discussing the optimality of the results for the first design problem, we follow the conjectures from (Bhasin et al., 2013). Here, $w_{n,t}$ represents the lowest weight of $CI(t)$ nonzero function of n variables. The conjecture was made in (Bhasin et al., 2013, Sec. C.2) that the values in each column of the Table 4 are non-decreasing. The values for $(n,t) \in \{(11,4-5)\}$ in Table 4 are interesting from this viewpoint: if the conjecture is true then they are optimal since they cannot be smaller than for $(n,t) \in \{(10,4-5)\}$, but the conjecture may be false; further investigations are needed to clarify this point. If $(n,t) \in \{(11,4-5)\}$ represent the minimal possible values; then since it is known from (Bhasin et al., 2013, Sec. C.1) that $w_{n,t} \geq 2w_{n-1,t-1}$, then the solution for $n = 12$ and $t \in \{5,6\}$ has also a minimal Hamming weight. Finally, for $n = 13$, by following the same reasoning, Hamming weights for $t \in \{6,7\}$ are again the optimal values, if that for $n = 11$ and $t = 4,5$ has. Actually, the value $w_{13,6} = 1\,024$ was already known (see Hedayat et al., 1999, Table 12.1, page 319), hence it does not appear as a boldface entry in Table 4.

When examining the results for the second design problem, we can again see that the GP outperforms all other algorithms. As in the first problem, the GA is not able to produce any competitive results already for the smallest size that equals 13 inputs. The results for the CGP are somewhat worse than expected if considering some of the results from related work (Picek et al., 2015a,c). However, those works consider only eight inputs Boolean functions so any comparison is hard to make. As in the first design problem, it seems that the CGP needs much more evaluations to compete with the GP on such large Boolean functions. Naturally, this should not pose a problem since CGP is much faster than the GP, but we refrain here from adding more evaluations in an effort to give as fair as possible comparison. When evaluating the best obtained solutions, it is not always easy to say which of the solutions are the best ones. The same fitness values are produced by various combinations of properties so that depends on the setting one considers. Furthermore, we see that even the nonlinearity and the algebraic degree values are often far from the theoretical best values. This is interesting since for smaller sizes (e.g. 8 inputs) previous results suggest that EC is highly competitive and is able to reach theoretical bounds for those properties.

We note that the most of the evolved functions have highly suboptimal values for the AI and FAI properties, which suggests that those properties must be parts of the fitness function. However, as already stated, their computational complexity is too high to be used in evolutionary experiments when considering the number of necessary evaluations. Naturally, this does not mean that some of the evolved solutions do not have good values of AI and FAI properties, but only that it is unrealistic to expect that all evolved solutions are highly fit with regards to those properties. Furthermore, if the evolutionary process creates a large number of individuals that must be evaluated a posteriori for those two properties, then one is again facing the same problem since the high computational complexity could prevent such an analysis. When considering the minimization of the number of terms in the ANF representation, we see that EAs are able to minimize it significantly. Therefore, the main drawback when using the EAs to evolve balanced Boolean functions with large number of inputs stems from the complexity of certain parts of the evaluation and not from some inability of EAs to work with such sizes of Boolean functions.

7 Conclusion

In this paper, we conducted an extensive analysis on the performance of EAs when evolving Boolean functions for cryptography. Furthermore, we consider only the Boolean function dimensions that are of practical relevance. In the first design problem, we evolve Boolean functions with minimal Hamming weight and different orders of CI property. By doing so, we are able to find the previously unknown values for sizes 11, 12, and 13. In our opinion, 16 inputs is approaching the upper limit of what is manageable for EAs and larger sizes would pose a problem. However, EAs should definitely present a viable option when generating Boolean functions, especially since most of the algebraic constructions are not suitable for the problem at hand.

When considering the second design problem, the situation becomes more complicated. Since our fitness functions are now more complex, the evolution process lasts longer and it is a question whether one can consider 16 inputs as a manageable size. Furthermore, two of the properties that are relevant (namely, AI and FAI) are computationally too complex to be parts of the fitness function.

Naturally, one can evolve Boolean functions and disregard those properties in the fitness, but later choose the best overall solutions by a posteriori evaluating those criteria. We investigated that approach by iteratively making our fitness function more complicated (i.e. consisting of more terms), but without significant success. If the quality of the AI and FAI properties is essential, then we believe that algebraic constructions pose a better option since they require a smaller number of evaluations.

We also emphasize that this is the first work, as far as we are aware, that considers the minimization of ANF representation as a design criterion. Since this property has a big impact on the implementation cost, we believe it is of high importance and should be considered in the future. When looking at the big picture, we conclude that EAs are powerful enough to evolve Boolean functions that are of practical dimensions for usages in cryptography. However, at least by following the representations and fitness functions we consider, 16 inputs also seems to be the close to the upper limit for EAs.

8 Acknowledgments

This work has been supported in part by Croatian Science Foundation under the project IP-2014-09-4882. In addition, this work was supported in part by the Research Council KU Leuven (C16/15/058) and IOF project EDA-DSE (HB/13/020).

References

Aguirre, H., Okazaki, H., and Fuwa, Y. (2007). An Evolutionary Multiobjective Approach to Design Highly Non-linear Boolean Functions. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO'07*, pages 749–756.

Bertoni, G., Macchetti, M., Negri, L., and Fragneto, P. (2004). Power-efficient ASIC Synthesis of Cryptographic Sboxes. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*, GLSVLSI '04, pages 277–281, New York, NY, USA. ACM.

Bhasin, S., Carlet, C., and Guilley, S. (2013). Theory of masking with codewords in hardware: low-weight *d*th-order correlation-immune Boolean functions. Cryptology ePrint Archive, Report 2013/303. `http://eprint.iacr.org/`.

Biham, E. and Shamir, A. (1991). Differential Cryptanalysis of DES-like Cryptosystems. In *Proceedings of the 10th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '90, pages 2–21, London, UK, UK. Springer-Verlag.

Burnett, L., Millan, W., Dawson, E., and Clark, A. (2004). Simpler methods for generating better Boolean functions with good cryptographic properties. *Australasian Journal of Combinatorics*, 29:231–247.

Burnett, L. D. (2005). *Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography*. PhD thesis, Queensland University of Technology.

Camion, P., Carlet, C., Charpin, P., and Sendrier, N. (1992). On Correlation-immune functions. In Feigenbaum, J., editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 86–100. Springer Berlin Heidelberg.

Canright, D. (2005). A Very Compact S-Box for AES. In Rao, J. R. and Sunar, B., editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer.

Canteaut, A. (2006). Open problems related to algebraic attacks on stream ciphers. In Ytrehus, O., editor, *Coding and Cryptography*, volume 3969 of *Lecture Notes in Computer Science*, pages 120–134. Springer Berlin Heidelberg.

Carlet, C. (2010). Boolean Functions for Cryptography and Error Correcting Codes. In Crama, Y. and Hammer, P. L., editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 257–397. Cambridge University Press, New York, NY, USA, 1st edition.

Carlet, C. (2013). A Survey on Nonlinear Boolean Functions with Optimal Algebraic Immunity Suitable for Stream Ciphers. *Vietnam Journal of Mathematics*, 41(4):527–541.

Carlet, C., Danger, J.-L., Guilley, S., and Maghrebi, H. (2012). Leakage Squeezing of Order Two. In Galbraith, S. and Nandi, M., editors, *Progress in Cryptology - INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 120–139. Springer Berlin Heidelberg.

Carlet, C. and Guilley, S. (2013). Side-channel Indistinguishability. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pages 9:1–9:8, New York, NY, USA. ACM.

Carlet, C. and Guilley, S. (2014). *Algebraic Curves and Finite Fields Cryptography and Other Applications*, chapter Correlation-immune Boolean functions for easing counter measures to side-channel attacks (chapter 3), pages 41–70. Radon Series on Computational and Applied Mathematics 16. De Gruyter.

Carlet, C. and Tang, D. (2015). Enhanced Boolean functions suitable for the filter model of pseudo-random generator. *Designs, Codes and Cryptography*, 76(3):571–587.

Carpi, R. B., Picek, S., Batina, L., Menarini, F., Jakobovic, D., and Golub, M. (2013). Glitch It If You Can: Parameter Search Strategies for Successful Fault Injection. In Francillon, A. and Rohatgi, P., editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 236–252. Springer.

Clark, A. J. (1998). *Optimisation heuristics for cryptology*. PhD thesis, Queensland University of Technology.

Clark, J. and Jacob, J. (2000). Two-Stage Optimisation in the Design of Boolean Functions. In Dawson, E., Clark, A., and Boyd, C., editors, *Information Security and Privacy*, volume 1841 of *Lecture Notes in Computer Science*, pages 242–254. Springer Berlin Heidelberg.

Clark, J. A., Jacob, J., Maitra, S., and Stănică, P. (2003). Almost Boolean functions: the design of Boolean functions by spectral inversion. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, volume 3, pages 2173–2180.

Clark, J. A., Jacob, J., and Stepney, S. (2004). Searching for cost functions. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1517–1524.

Clark, J. A., Jacob, J. L., Stepney, S., Maitra, S., and Millan, W. (2002). Evolving Boolean Functions Satisfying Multiple Criteria. In *Progress in Cryptology - INDOCRYPT 2002*, pages 246–259.

Courtois, N. (2003). Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In Boneh, D., editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer Berlin Heidelberg.

Courtois, N. and Meier, W. (2003). Algebraic Attacks on Stream Ciphers with Linear Feedback. In Biham, E., editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer Berlin Heidelberg.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Detombe, J. and Tavares, S. E. (1992). Constructing Large Cryptographically Strong S-boxes. In Seberry, J. and Zheng, Y., editors, *Advances in Cryptology - AUSCRYPT '92, Workshop on the Theory and Application of Cryptographic Techniques, Gold Coast, Queensland, Australia, December 13-16, 1992, Proceedings*, volume 718 of *Lecture Notes in Computer Science*, pages 165–181. Springer.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

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

Grassl, M. (2007). Bounds on the minimum distance of linear codes and quantum codes. Online available at `http://www.codetables.de/`. Accessed on 2012-07-23.

Guo-Zhen, X. and Massey, J. (1988). A spectral characterization of correlation-immune combining functions. *IEEE Transactions on Information Theory*, 34(3):569–571.

Hawkes, P. and Rose, G. (2004). Rewriting Variables: The Complexity of Fast Algebraic Attacks on Stream Ciphers. In Franklin, M., editor, *Advances in Cryptology CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 390–406. Springer Berlin Heidelberg.

Hedayat, A. S., Sloane, N. J. A., and Stufken, J. (1999). *Orthogonal Arrays, Theory and Applications*. Springer series in statistics. Springer, New York. ISBN 978-0-387-98766-8.

Hernandez-Castro, J. C., Estévez-Tapiador, J. M., Garnacho, A. R., and Ramos-Alvarez, B. (2006). Wheedham: An Automatically Designed Block Cipher by means of Genetic Programming. In *IEEE International Conference on Evolutionary Computation, CEC 2006, part of WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 192–199.

Heys, H. M. and Tavares, S. E. (1994). The Design of Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis. In Denning, D. E., Pyle, R., Ganesan, R., and Sandhu, R. S., editors, *CCS '94, Proceedings of the 2nd ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 2-4, 1994.*, pages 148–155. ACM.

Hrbacek, R. and Dvorak, V. (2014). Bent Function Synthesis by Means of Cartesian Genetic Programming. In Bartz-Beielstein, T., Branke, J., Filipič, B., and Smith, J., editors, *Parallel Problem Solving from Nature - PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 414–423. Springer International Publishing.

Izbenko, Y., Kovtun, V., and Kuznetsov, A. (2008). The design of boolean functions by modified hill climbing method. Cryptology ePrint Archive, Report 2008/111.

Jakobovic, D. (2014). Evolutionary Computation Framework. `http://ecf.zemris.fer.hr/`.

Kavut, S. and Yücel, M. (2003). Improved Cost Function in the Design of Boolean Functions Satisfying Multiple Criteria. In Johansson, T. and Maitra, S., editors, *Progress in Cryptology - INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 121–134. Springer Berlin Heidelberg.

Lamenca-Martinez, C., Hernandez-Castro, J. C., Estevez-Tapiador, J. M., and Ribagorda, A. (2006). Lamar: A New Pseudorandom Number Generator Evolved by Means of Genetic Programming. In Runarsson, T. P., Beyer, H.-G., Burke, E., Merelo-Guervós, J., Whitley, L., and Yao, X., editors, *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 850–859. Springer Berlin Heidelberg.

MacWilliams, F. J. and Sloane, N. J. A. (1977). *The Theory of Error-Correcting Codes*. Elsevier, Amsterdam, North Holland. ISBN: 978-0-444-85193-2.

Mangard, S., Oswald, E., and Popp, T. (2007). *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Mariot, L. and Leporati, A. (2015a). 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*, pages 33–45.

Mariot, L. and Leporati, A. (2015b). 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*, pages 1425–1426.

Massey, J. (1969). Shift-register synthesis and BCH decoding. *Information Theory, IEEE Transactions on*, 15(1):122–127.

Massey, J. L. (1995). Some Applications of Coding Theory in Cryptography. In *Codes and Ciphers: Cryptography and Coding IV*, pages 33–47.

Matsui, M. and Yamagishi, A. (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, pages 81–91, Berlin, Heidelberg. Springer-Verlag.

McLaughlin, J. and Clark, J. A. (2013). Evolving balanced Boolean functions with optimal resistance to algebraic and fast algebraic attacks, maximal algebraic degree, and very high nonlinearity. Cryptology ePrint Archive, Report 2013/011.

Meier, W., Pasalic, E., and Carlet, C. (2004). Algebraic Attacks and Decomposition of Boolean Functions. In Cachin, C. and Camenisch, J., editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer Berlin Heidelberg.

Meier, W. and Staffelbach, O. (1988). Fast Correlation Attacks on Stream Ciphers. In Barstow, D., Brauer, W., Brinch Hansen, P., Gries, D., Luckham, D., Moler, C., Pnueli, A., Seegmüller, G., Stoer, J., Wirth, N., and Günther, C., editors, *Advances in Cryptology - EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. Springer Berlin Heidelberg.

Millan, W., Clark, A., and Dawson, E. (1997). An Effective Genetic Algorithm for Finding Highly Nonlinear Boolean Functions. In *Proceedings of the First International Conference on Information and Communication Security*, ICICS '97, pages 149–158, London, UK, UK. Springer-Verlag.

Millan, W., Clark, A., and Dawson, E. (1998). Heuristic design of cryptographically strong balanced Boolean functions. In *Advances in Cryptology - EUROCRYPT '98*, pages 489–499.

Millan, W., Clark, A., and Dawson, E. (1999). Boolean Function Design Using Hill Climbing Methods. In Pieprzyk, J., Safavi-Naini, R., and Seberry, J., editors, *Information Security and Privacy*, volume 1587 of *Lecture Notes in Computer Science*, pages 1–11. Springer Berlin Heidelberg.

Millan, W., Fuller, J., and Dawson, E. (2004). New concepts in evolutionary search for Boolean functions in cryptology. *Computational Intelligence*, 20(3):463–474.

Miller, J. F. (1999). An Empirical Study of the Efficiency of Learning Boolean Functions using a Cartesian Genetic Programming Approach. In Banzhaf, W., Daida, J. M., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M. J., and Smith, R. E., editors, *GECCO*, pages 1135–1142. Morgan Kaufmann.

Miller, J. F., editor (2011). *Cartesian Genetic Programming*. Natural Computing Series. Springer Berlin Heidelberg.

Paar, C. and Pelzl, J. (2010). *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer.

Picek, S., Batina, L., and Jakobovic, D. (2014a). Evolving DPA-Resistant Boolean Functions. In *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, pages 812–821.

Picek, S., Carlet, C., Jakobovic, D., Miller, J. F., and Batina, L. (2015a). 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*, pages 1095–1102.

Picek, S., Ege, B., Batina, L., Jakobovic, D., Chmielewski, L., and Golub, M. (2014b). 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*, CS2 '14, pages 13–18, New York, NY, USA. ACM.

Picek, S., Guilley, S., Carlet, C., Jakobovic, D., and Miller, J. F. (2015b). 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*, pages 71–82.

Picek, S., Jakobovic, D., and Golub, M. (2013). Evolving Cryptographically Sound Boolean Functions. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, pages 191–192, New York, NY, USA. ACM.

Picek, S., Jakobovic, D., Miller, J. F., Batina, L., and Cupic, M. (2016). Cryptographic Boolean functions: One output, many design criteria. *Appl. Soft Comput.*, 40:635–653.

Picek, S., Jakobovic, D., Miller, J. F., Marchiori, E., and Batina, L. (2015c). Evolutionary Methods for the Construction of Cryptographic Boolean Functions. In *Genetic Programming - 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, pages 192–204.

Picek, S., Marchiori, E., Batina, L., and Jakobovic, D. (2014c). Combining Evolutionary Computation and Algebraic Constructions to Find Cryptography-Relevant Boolean Functions. In *Parallel Problem Solving from Nature - PPSN XIII - 13th International Conference, Ljubljana, Slovenia, September 13-17, 2014. Proceedings*, pages 822–831.

Poli, R., Langdon, W. B., and McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`. (With contributions by J. R. Koza).

Preneel, B., Van Leekwijck, W., Van Linden, L., Govaerts, R., and Vandewalle, J. (1991). Propagation characteristics of Boolean functions. In *Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, EUROCRYPT '90, pages 161–173, New York, NY, USA. Springer-Verlag New York, Inc.

Rodier, F. (2006). Asymptotic Nonlinearity of Boolean Functions. *Designs, Codes and Cryptography*, 40(1):59–70.

Rønjom, S. and Helleseth, T. (2007). A New Attack on the Filter Generator. *Information Theory, IEEE Transactions on*, 53(5):1752–1758.

Seberry, J., Zhang, X., and Zheng, Y. (1993). Systematic Generation of Cryptographically Robust S-Boxes. In Denning, D. E., Pyle, R., Ganesan, R., Sandhu, R. S., and Ashby, V., editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 171–182. ACM.

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

Siegenthaler, T. (1985). Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Trans. Comput.*, 34(1):81–85.

Siegenthaler, T. (2006). Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications (Corresp.). *IEEE Transactions on Information Theory*, 30(5):776–780.

Youssef, A. M. and Tavares, S. E. (1995). Resistance of Balanced s-Boxes to Linear and Differential Cryptanalysis. *Inf. Process. Lett.*, 56(5):249–252.