Applied Soft Computing xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Cryptographic Boolean functions: One output, many design criteria

Q1 Stjepan Picek[a,b,*], Domagoj Jakobovic[b], Julian F. Miller[c], Lejla Batina[d], Marko Cupic[b]

[a] KU Leuven, ESAT/COSIC and iMinds, Kasteelpark Arenberg 10, bus 2452, B-3001 Leuven, Heverlee, Belgium
Q2 [b] Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia
[c] Department of Electronics, University of York, UK
[d] Digital Security Group, Radboud University, The Netherlands

ARTICLE INFO

Article history:
Received 17 June 2015
Received in revised form 3 October 2015
Accepted 29 October 2015
Available online xxx

Keywords:
Evolutionary algorithms
Boolean functions
Cryptography
Comparison
Analysis

ABSTRACT

Boolean functions represent an important primitive in the design of various cryptographic algorithms. There exist several well-known schemes where a Boolean function is used to add nonlinearity to the cipher. Thus, methods to generate Boolean functions that possess good cryptographic properties present an important research goal. Among other techniques, evolutionary computation has proved to be a well-suited approach for this problem. In this paper, we present three different objective functions, where each inspects important cryptographic properties of Boolean functions, and examine four evolutionary algorithms. Our research confirms previous results, but also sheds new insights on the effectiveness and comparison of different evolutionary algorithms for this problem.

1. Introduction

Q5 In the last few decades there has been significant research on Boolean functions, for cryptography as well for other uses like algebraic coding and sequence design [1–4]. Accordingly, there exist many different approaches for the construction of Boolean functions as well as numerous rationales behind the choice of properties relevant for such functions. Although Boolean functions in cryptography have a less prominent position than 20 or more years ago, they still represent an important cryptographic primitive. In this paper, we concentrate on the use of Boolean functions as building blocks in filter and combiner generators [4].

Boolean functions are often the only nonlinear element in stream ciphers and without them a cipher would be trivial to break. Therefore, it is not surprising that there exists a substantial body of work on methods of generating Boolean functions. However, such applications of Boolean functions in cryptography are not the only ones. For instance, they can be also used to resist side-channel attacks. When discussing side-channel countermeasures, one important class contains 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 that can be substantial in embedded devices or smart cards. It has been shown that correlation immune Boolean functions that have minimal Hamming weight reduce the masking overhead [5,6]. However, most of the algebraic constructions are designed to produce balanced or bent Boolean functions and are therefore not suitable for this task. Consequently, it would be beneficial to have some other method of constructing Boolean functions.

We distinguish between three main approaches for generating Boolean functions for cryptographic usages: algebraic constructions, random generation and heuristic constructions (and various combinations of these approaches) [7]. Algebraic constructions use some mathematical procedure to create a Boolean function with good cryptographic properties. One example of such a construction is the Maiorana-McFarland construction [4]. Random generation of Boolean functions also has its strong points, the most prominent being that it is easy and fast. However, the resulting Boolean functions usually have suboptimal properties for cryptographic usages [8]. Heuristic methods offer an easy and efficient way of producing a large number of Boolean functions with very good cryptographic properties [2]. Among different heuristic approaches, evolutionary computation (EC) and more specifically evolutionary algorithms (EAs) offer highly competitive results when generating Boolean functions for cryptography [9,10]. It is worth mentioning that EAs can be used either as the primary or the secondary construction method. In primary constructions one obtains new functions without using known ones. In secondary constructions, one uses

Q3 * Corresponding author at: KU Leuven, ESAT/COSIC and iMinds, Kasteelpark Arenberg 10, bus 2452, B-3001 Leuven, Heverlee, Belgium. Tel.: +385 98226407.
E-mail address: stjepan@computer.org (S. Picek).

http://dx.doi.org/10.1016/j.asoc.2015.10.066
1568-4946/© 2015 Elsevier B.V. All rights reserved.

already known Boolean functions to construct new Boolean functions (either with different properties or sizes) [4].

In this paper, we experiment with several different EAs in order to explore their efficiency in the evolution of Boolean functions with properties necessary for use in cryptography. More precisely, we investigate Genetic Algorithms (GAs), Genetic Programming (GP), Cartesian Genetic Programming (CGP), and Evolution Strategies (ES). Furthermore, instead of concentrating on only one objective, we investigate three objectives, represented with different fitness functions. Since we experiment with several methods and fitness functions, it is not feasible to conduct experiments with all possible parameter combinations. Therefore, we restrict our attention to combinations which, based on our previous results and usual settings for those algorithms, provide acceptable results.

Since an investigation of any one of the four aforementioned algorithms on a single objective could easily constitute a whole paper, this work should be regarded as a practical guide and not as an in-depth analysis. However, it is worth noting that we perform more than 30,000 independent experimental runs for various EAs in order to conduct statistical analysis. We concentrate here only on Boolean functions with eight inputs. Eight inputs is a relatively small size for Boolean functions for stream ciphers (it is commonly considered that 13 is a strict minimum for resisting algebraic attacks), but we still believe it is an interesting case. Indeed, for instance ciphers RAKAPOSHI [11] and Achterbahn [12] use Boolean functions of that size. Evolving Boolean functions is a challenging task because there exist 2^{2^n} possible Boolean functions of n inputs (for eight inputs this gives 2^{256} candidate solutions). Therefore, with anything more than five inputs it is impossible to do an exhaustive search. Furthermore, with larger sizes of Boolean functions the search space grows, but also the computational complexity of their various properties grows. Thus, it is unrealistic to expect that EAs can work for much larger sizes than 13 inputs. This is because the computation of some properties like the algebraic immunity, and even more the fast algebraic immunity, becomes rapidly prohibitive for large numbers of inputs.

The rest of this paper is organized as follows. In Section 2, we present several applications of stochastic algorithms when generating Boolean functions appropriate for cryptographic usages. Section 3 presents relevant representations and cryptographic properties of Boolean functions. Next, in Section 4 we give fitness functions used in our experiments. Section 5 deals with the experimental setup and EAs we use. In Section 6, we give results for each of the objective functions as well as a short discussion on the results. Finally, we end with a short summary in Section 7.

2. Related work

As noted, there have been many applications of heuristic methods for the generation of Boolean functions for cryptographic usages. Here, we describe previous work directly related to our investigation in a chronological order. As far as the authors know, the first application of GAs to the evolution of cryptographically suitable Boolean functions emerged in 1997 when Millan et al. experimented with GAs to evolve Boolean functions with high nonlinearity [13]. In his thesis, Clark presented several applications of optimization techniques in the field of cryptology [14]. One of the applications is the evolution of Boolean functions with high nonlinearity using GA and hill climbing techniques. Millan, Clark, and Dawson used GAs to evolve Boolean functions that have high nonlinearity [15]. In conjunction with the GA they used hill climbing together with a resetting step in order to find Boolean functions with even higher nonlinearity for sizes of up to 12 inputs. More specifically, when discussing Boolean functions with eight inputs, they found balanced functions with nonlinearity 112 and correlation immunity equal to one.

Millan et al. used variations of a hill climbing method in order to find Boolean functions that have high nonlinearity and low autocorrelation [16].

Clark and Jacob experimented with two-stage optimization to generate Boolean functions with high nonlinearity and low autocorrelation [17]. They used a combination of simulated annealing (SA) and hill climbing with a cost function motivated by Parseval theorem. Clark et al. used SA to generate Boolean functions with cryptographically relevant properties where they considered balanced function with high nonlinearity and with the correlation immunity less and equal to two [18].

Kavut and Yücel developed an improved cost function for a search that combines SA and hill climbing [19]. In their approach, the authors were able to find some functions of eight and nine inputs that have a combination of nonlinearity and autocorrelation values previously unattained. They also experimented with a three-stage optimization method that combines SA and two hill climbing algorithms with different objectives.

Clark et al. experimented with SA in order to design Boolean functions using spectral inversion [20]. They observed that many cryptographic properties of interest are defined in terms of the Walsh–Hadamard transform values. Therefore, they worked in the spectral domain where the cost function punishes those solutions that are not Boolean functions. More precisely, on the basis of Parseval's theorem one can infer what values should be in a Walsh–Hadamard spectrum, but it is impossible to say what the positions should be. Therefore, when generating a Walsh–Hadamard spectrum it is necessary to make an inverse transform to verify that the spectrum indeed maps to a Boolean function. Burnett et al. presented two heuristic methods where the goal of the first method was to generate balanced Boolean functions with high nonlinearity and low autocorrelation. The second method aimed to generate resilient functions with high nonlinearity and algebraic degree that maximizes the Siegenthaler inequality [21]. Millan, Fuller and Dawson proposed a new adaptive strategy for a local search algorithm for the generation of Boolean functions with high nonlinearity [8]. Additionally, they introduced the notion of the graph of affine equivalence classes of Boolean functions. Burnett in her thesis used three heuristic techniques to evolve Boolean functions [2]. The first method aimed to evolve balanced functions with high nonlinearity. The second method was used to find balanced Boolean functions with high nonlinearity that are correlation immune. The last method was used to find balanced functions with high nonlinearity and propagation characteristics different from zero. Aguirre et al. used a multi-objective random bit climber to search for balanced Boolean functions of size up to eight inputs that have high nonlinearity [22]. Their results indicate that the multi-objective approach is highly efficient when generating Boolean functions that have high nonlinearity. Izbenko et al. used a modified hill climbing algorithm to transform bent functions to balanced Boolean functions with high nonlinearity [23]. McLaughlin and Clark experimented with SA to generate Boolean functions that have optimal values of algebraic immunity, fast algebraic resistance, and algebraic degree [24]. In their work, they experimented with Boolean functions with sizes of up to 16 inputs.

Picek, Jakobovic, and Golub experimented with GA and GP to find Boolean functions that possess several optimal properties [9]. As far as the authors know, this is the first application of GP for evolving cryptographically suitable Boolean functions. Hrbacek and Dvorak used CGP to evolve bent Boolean functions of sizes up to 16 inputs [25] where the authors experimented with several configurations of algorithms in order to speed up the evolution process. They did not limit the number of generations and therefore they succeeded in finding bent function in each run for sizes between