in Proceedings of 1st Balcan Conference on Informatics,
November 2003, Thessaloniki, Greece

Generating highly nonlinear Boolean functions using a genetic algorithm

A.Dimovski[1], D.Gligoroski[2],

Abstract: **In this paper a few algorithms are presented which assist with finding Boolean functions with good cryptographic properties, especially with high nonlinearity. First, a basic hill-climbing algorithm is described which improve the nonlinearity of a Boolean function. Then this algorithm is modified to incorporate a genetic algorithm. It is shown that these new search techniques are extremely powerful when compared to traditional random search techniques. Experimental results successfully prove this statement.**

Keywords: **Boolean functions, nonlinearity, Hill Climbing Algorithm, Genetic algorithm**

I. Introduction

In this paper, we will present a useful application of the genetic algorithm to the field of cryptography. The genetic algorithm is used to search for cryptographically sound Boolean functions. Most block and stream ciphers incorporate Boolean functions, which are chosen to satisfy a number of cryptographic criteria.

There are many cryptographic properties of Boolean functions, and some of them will be described in the next section. In this paper, the property of nonlinearity is considered, although the work could be extended to include other cryptographic properties. When designing cryptosystems (ciphers) careful consideration must be given to the choice of functions used. High nonlinearity is an extremely important property required in order to reduce the effectiveness of attacks such as linear cryptoanalysis – proposed by Matsui.

II. Boolean functions and their cryptographic properties

In this section, we will describe Boolean functions, their representation, operators and properties.

The most basic representation of a Boolean function is by its binary truth table. The binary truth table of a Boolean function of n variables is denoted $f(x)$ where $f(x) \in \{0, 1\}$ and $x = \{x_1, x_2, \ldots, x_n\}$, $x_i \in \{0, 1\}$, $i = 1, \ldots, n$. The truth table contains 2^n elements corresponding to all possible combinations of the n binary inputs.

Sometimes it is desirable to consider a Boolean function over the set {1, -1} rather than {0, 1}. The polarity truth table of a Boolean function is denoted $f\^(x)$ where $f\^(x) \in \{1, -1\}$ and $f\^(x) = (-1)^{f(x)} = 1 - 2f(x)$. So, if $f(x) = 1$ then $f\^(x) = -1$, and if $f(x) = 0$ then $f\^(x) = 1$. It is also important to note that XOR over {0, 1} is equivalent to real multiplication over {-1, 1}. Thus,

$$h(x) = f(x) \oplus g(x) =$$
$$\Rightarrow \quad h\^(x) = f\^(x) \cdot g\^(x)$$

Two fundamental properties of Boolean functions are Hamming weight and Hamming distance. The Hamming weight of a Boolean function is the number of ones in the binary truth table, or equivalently the number of -1s in the polarity truth table. So, the Hamming weight of a Boolean function f, $hwt(f)$, is given by:

$$hwt(f) = \sum_x f(x) = \frac{1}{2}(2^n - \sum_x f\^(x)) \qquad (1)$$

The Hamming distance between two Boolean functions is the number of positions in which their truth tables differ. The Hamming distance between two Boolean functions, $dist(f, g)$, can be calculated as follows:

$$dist(f,g) = \sum_x (f(x) \oplus g(x)) = \frac{1}{2}(2^n - \sum_x f\^(x)) \quad (2)$$

How well two Boolean functions correlate is also of interest. The correlation between two Boolean functions, $c(f,g)$, gives an indication of the extent to which two functions approximate each other. The correlation is a real number in the range [-1, 1], and is given by:

$$c(f,g) = 1 - \frac{dist(f,g)}{2^{n-1}} = 2^{-n} \sum_x f\^(x) g\^(x) \qquad (3)$$

A linear function, $Lw(x)$, where $w \in Z_2^n$, is defined by:

$$Lw(x) = w \cdot x = w_1 x_1 \oplus w_2 x_2 \oplus \ldots \oplus w_n x_n$$

[1] Faculty of Natural Sciences and Mathematics, Ss. Cyril and Methodius University Arhimedova b.b., PO Box 162, 1000 Skopje, Macedonia adimovski@ii.edu.mk
[2] Faculty of Natural Sciences and Mathematics, Ss. Cyril and Methodius University Arhimedova b.b., PO Box 162, 1000 Skopje, Macedonia gligoroski@yahoo.com

An affne function is one of the form:

$$Aw(x) = w \cdot x \oplus c$$

where $c \in Z_2$.

The Hamming distance to linear functions is an important cryptographic property, since ciphers that employ nearly linear functions can be broken easily by a variety of methods. So, we get the definition of a new cryptographic property of Boolean functions, nonlinearity. The nonlinearity of a Boolean function is the minimum distance to any affine function. In order to determine the nonlinearity of a Boolean function, we should find Walsh - Hadamard Transform, WHT, of that Boolean function:

$$F\hat{}(w) = \sum_x f\hat{}(x) L\hat{}\, w(x) \quad (4)$$

It is clear from this definition that the value of $F\hat{}(w)$ is closely related to the Hamming distance between $f(x)$ and the linear function $Lw(x)$, in fact $c(f, Lw) = F\hat{}(w) / 2^n$.

The nonlinearity, N_f, of f is raleted to the maximum magnitude of WHT values, WH_{MAX}, and is given by:

$$N_f = \frac{1}{2} \cdot (2^n - WH_{MAX}) \quad (5)$$

Clearly in order to increase the nonlinearity of a Boolean function, WH_{MAX} must be decreased. A function is uncorrelated with linear function $Lw(x)$ when $F\hat{}(w) = 0$. Cryptographically, it would be desirable to find Boolean functions, which have all WHT values equal to zero, since such functions have no correlation to any affine functions. However, it is known that such functions do not exist. In [4], there is a theorem, which states that the sum of the squares of the WHT values is the same constant for every Boolean function: $\sum_w F\hat{}^2(w) = 2^{2n}$. So, there is an opportunity only to minimize affine correlation, and in that way to maximize the nonlinearity of functions.

It is known that the Bent functions [6] satisfy the property that $|F\hat{}(w)| = 2^{n/2}$ for all w. Bent functions exist only for even n, and they attain the maximum possible nonlinearity of $N_{BENT} = 2^{n-1}-2^{(n-1)/2}$. It is an open problem to determine an expression for the maximum nonlinearity of functions with an odd number of inputs. It is known that, for n odd, it is possible to construct a function with nonlinearity $2^{n-1}-2^{(n-1)/2}$ by concatenating Bent functions. Still, it is shown that for $n = 15$, this value is not the upper bound of the nonlinearity.

III. Improving Nonlinearity

Now, we will describe a technique, which enables the creation of a complete list of Boolean function inputs such that complementing any one of the corresponding truth table positions will increase the nonlinearity of the function. This list of truth table positions is referred to as the 1-Improvement Set of *f*, or *1-ISf* for short. A formal definition of the *1-ISf* is:

Definition 1. Let $g(x) = f(x) \oplus 1$ for x = x_a, and $g(x) = f(x)$ for all other x. If $Ng > Nf$ then $x_a \in 1 - ISf$.

The set *1-ISf* may be empty in which case *f* is referred to as 1-locally maximum for nonlinearity and cannot be improved using the technique described below. Since all Bent functions are globally maximum, their 1-Improvement Sets must be empty. It is computationally intensive to exhaustively alter truth table positions, find new WHT values and determine the set *1-ISf*. In this section a set of conditions are presented which provide a method of determining whether or not an input *x* is in the 1-Improvement Set.

In order to find the *1-ISf* of a Boolean function it is first necessary to find values of the WHT coefficients with the Equation 4.

Definition 2. Let *f* be a Boolean function with Walsh - Hadamard transform $F\hat{}(w)$, where WH_{MAX} denotes the maximum absolute value of $F\hat{}(w)$. There will exist one or more linear functions $Lw(x)$ that have minimum distance to *f*, and $|F\hat{}(w)| = WH_{MAX}$ for these w. The following sets are defined:

$$W^+_1 = \{w: F\hat{}(w) = WH_{MAX}\}$$
$$W^-_1 = \{w: F\hat{}(w) = -WH_{MAX}\}$$

Also needed are the sets of w, for which the WHT magnitude is close to the maximum WH_{MAX}:

$$W^+_2 = \{w: F\hat{}(w) = WH_{MAX} - 2\}$$
$$W^-_2 = \{w: F\hat{}(w) = -(WH_{MAX} - 2)\}$$

When a truth table is changed in exactly one place, all WHT values are changed by +2 or -2. So, in order to increase the nonlinearity of a function, the WHT values in set W^+_1 must change by -2, the WHT values in set W^-_1 must change by +2, and also the WHT values in W^+_2 must change by –2 and the WHT values in W^-_2 must change by +2. The first two conditions are obvious, and the second two conditions are required so that all other $|F\hat{}(w)|$ remain less than WH_{MAX}.

Theorem 1. Given a Boolean function *f* with WHT $F\hat{}(w)$, and define sets $W^+ = W^+_1 \cap W^+_2$ and $W^- = W^-_1 \cap W^-_2$. For an input *x* to be an element of the Improvement Set *1-ISf*, the following two conditions must be satisfied:

(i) $f(x) = Lw(x)$ *for all* $w \in W^+$, *and*

(ii) $f(x) \neq Lw(x)$ *for all* $w \in W^-$.

Proof: Let's start by considering the conditions to make WHT values change by a desired amount. When $F\hat{}(w)$ is positive, there are more 1 than -1 in the polarity truth table, and more 0 than 1 in the binary truth table of $f(x) \oplus Lw(x)$. Thus changing a single 0, in the truth table of $f(x) \oplus Lw(x)$, to a 1 will make $\Delta F(w) = -2$. This means that an input x, is selected such that $f(x) = Lw(x)$. A change of -2 is desired for all WHT values with $w \in W^+$, and this proves condition (i). A similar argument proves condition (ii).

The following theorem shows how to modify the WHT values of a Boolean function that has been altered in a single truth table position.

Theorem 2. Let *g(x)* be obtained from *f(x)* by complementing the output for a single input x_a. Then each component of the WHT values of *g(x)*, $G^\wedge(w) = F^\wedge(w) + \Delta(w)$, can be obtained as follows: If $f(x_a) = Lw(x_a)$, then $\Delta(w) = -2$, else $\Delta(w) = +2$. **Proof:** If $f(x_a) = Lw(x_a)$, then $f^\wedge(x_a) \cdot L^\wedge w(x_a) = 1$, and this 1 contributes to the sum in $F^\wedge(w)$. Changing the value of $f(x_a)$ changes this contribution to -1, so $\Delta F^\wedge(w) = -2$. Similarly if $f(x) \neq Lw(x)$, then $\Delta F^\wedge(w) = +2$.

IV. Hill Climbing Algorithm

In this section, we will describe the one step improvement algorithm Hill Climbing 1.

The one step improvement algorithm, Hill Climbing 1, takes as its input the binary truth table of a Boolean function and corresponding WHT values, and recursively improves the Boolean function's nonlinearity until the function is 1-locally maximum, and that case its 1-Improvement Set is empty. The Hill Climbing 1 algorithm tries each bit in the truth table successively in an attempt to find a candidate bit that, upon complementation, will improve the function's nonlinearity by one. The algorithm terminates when no improvement in nonlinearity can be obtained by complementing any one of the bit in the function's truth table. Description of the Hill Climbing 1 algorithm is given:

1. The algorithm is given the binary truth table of a Boolean function BF and corresponding WHT values.
2. Determine the maximum WHT value - WH_{MAX}.
3. By parsing the WHT values, we find those *w* which correspond to WHT or $F^\wedge(w)$ values equal to $|WH_{MAX}|$ and $|WH_{MAX} - 2|$. In this manner create the two sets: $W^+ = W^+_1 \cap W^+_2$ and $W^- = W^-_1 \cap W^-_2$.
4. For i = 1, . . . 2n, do:
   (a) Check whether the i-th bit in the truth table of BF, satisfy conditions *(i)* and *(ii)* of Theorem 1, for the sets W^+ and W^-.
   (b) If conditions are satisfied, then first complement i-th bit in the truth table of BF to produce the new Boolean function BF', and calculate the updated WHT' values using Theorem 2, finally restart the algorithm, i.e. return back to the Step 1 with new arguments BF' and WHT'.
5. BF represents a 1-locally maximum Boolean function and the algorithm is finished.

V. Using a Genetic Algorithm to the Search

In this section, the genetic algorithm is used to improve the Hill Climbing 1 algorithm described above. Solution, in this Genetic algorithm, is a Boolean function, which will be represented as a binary string. In this case the binary string represents the binary truth table of the Boolean function. Given a solution representation, there are three other requirements of the genetic algorithm, namely a solution evaluation technique, reproduction and mutation operations.

The genetic algorithm requires a method of assessing and comparing solutions. The fitness, which is used here, is simply the nonlinearity N_f of the Boolean function. In other words, one solution is better than other solution if the first one *f* has higher nonlinearity N_f.

The genetic algorithm also requires a method for combining two solutions (parents) in order to obtain new solutions (children). Here, we will use a reproduction operation called 'merging', which is described below.

Given the binary truth tables of two Boolean functions f_1 and f_2 of *n* variables and Hamming distance *d*. The 'merge' operation is defined as:

- If $d \leq 2^{n-1}$
  MERGE f_1, $f2$ (x) = $f_1(x)$, if $f_1(x) = f_2(x)$
  a random bit, otherwise
- Else
  MERGE f_1, $f2$ (x) = $f_1(x)$, if $f_1(x) \neq f_2(x)$
  a random bit, otherwise

This 'merge' operation includes implicit mutation. Since random mutation of a highly nonlinear function is likely to reduce the nonlinearity, additional mutations are avoided and instead the merge is relied upon to direct the pool into new areas of the search space. The motivation for this operation is that two functions that are highly nonlinear and close to each other will be close to some local maximum, and the merging operation produces a function also in the same region, hopefully close to that maximum. Also when applied to uncorrelated functions, the merge operation produces children spread over a large area, thus allowing the genetic algorithm to search the space more fully.

Combining each of the genetic algorithm operations described above the overall algorithm is obtained. Generally the initial solution pool is generated randomly, and this is acceptable since very few randomly generated functions have low nonlinearity. The problem with random generation is that very highly nonlinear functions are difficult to find.

In this genetic algorithm, all possible combinations of parents undergo the recombination process. If the pool size is P, then there are P(P-1)/2 such pairings. We should initialize the following algorithm parameters: the maximum number of iterations that the algorithm should perform *MAX*, the size of the solution pool *P*, and *HC* is Boolean value indication whether or not the algorithm should incorporate the Hill Climbing 1 algorithm. Description of the genetic algorithm used to generate highly nonlinear functions is given:

1. Initialize the algorithm parameters: *MAX*, *P*, and *HC*.
2. Generate a pool of *P* random Boolean functions and calculate their corresponding WHT values.
3. For i = 1, . . . 2^n, do:
   (a) For each possible pairing of the functions in the solution pool, do:

- Perform the 'merge' operation on the two parents to produce a child – solution
- If *HC* = true, call Hill Climbing 1 function for the child
- If the resulting child is not already in the list of children, add the new child to the list of children.

(b) Select the best solutions from the list of children and the current pool of solutions.

4. Output the best solution from the current solution pool.

VI. Experimental Results

The results presented in this section illusstrate the merits of the genetic algorithm search over both random Boolean function generation and the Hill Climbing 1 algorithm.

As a benchmark the best results obtained from random search for functions with inputs ranging from eight *n* = 8 to twelve *n* = 12 were obtained for various search sizes from 100 to 100000 functions. The results for nonlinearity of this extensive search are given in Table 1. The table clearly shows that increasing the sample size 10 times only marginally increases the nonlinearity obtained. These results confirm the property of Boolean functions: most functions do not have low linearity, but very highly nonlinear functions are extremely rare.

TABLE 1
BEST NONLINEARITY ACHIEVED BY RANDOM SEARCH

Sample size	8	9	10	11	12
100	110	228	469	958	1946
1000	111	228	470	959	1947
10000	111	229	470	960	1949
100000	112	229	470	960	1950

The acronyms used in Tables 2 and 3 have the following meanings: RHC means a random search utilizing the Hill Climbing 1 algorithm, GA means a basic genetic algorithm with no hill climbing, GA HC means a genetic algorithm which incorporate the Hill Climbing 1 algorithm.

Tables 2 and 3 indicate the best results achived by the algorithms when they are forced to terminate after a specific number of functions have been tested. A direct comparison between random generation with Hill Climbing 1 algorithm, and a simple genetic algorithm without hill climbing shows that these algorithms are about equally effective for 100 and 1000 function tests. Other experiments have suggested that as the computation bound is increased, the performance of the genetic algorithm will eventually exceed that of RHC. It is interesting to note that the best algorithm is clearly a genetic algorithm with hill climbing. This hybrid algorithm is able to quickly obtain functions far better than the benchmarks.

TABLE 2
BEST NONLINEARITY ACHIEVED AFTER TESTING 100 FUNCTIONS

Method	8	9	10	11	12
Benchmark	110	228	469	958	1946
R HC	112	232	475	964	1958
GA	111	229	470	959	1951
GA HC	113	232	474	968	1962

TABLE 3
BEST NONLINEARITY ACHIEVED AFTER TESTING 1000 FUNCTIONS

Method	8	9	10	11	12
Benchmark	111	228	470	959	1947
R HC	112	232	476	966	1960
GA	113	232	475	964	1956
GA HC	114	236	480	974	1970

V. References

[1] K.G. Beauchamp. Applications of Walsh and Related Functions. Academic Press, 1984.

[2] D.E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading, Massechusetts, 1989.

[3] J.Dj. Golic. Linear Cryptanalysis of Stream Ciphers. In Fast Software Encryption, 1994 Leuven Workshop, LNCS, volume 1008, pages 154169, December 1994.

[4] W. Meier and O. Staffelbach. Nonlinearity Criteria for Cryptographic Functions. In Advances in Cryptology - Eurocrypt 89, Proceedings, LNCS, volume 434, pages 549562. Springer-Verlag, 1990.

[5] O.S. Rothaus. On Bent Functions. Journal of Combinatorial Theory (A), 20:300 305, 1976.

[6] William Millan, Andrew Clark, and Ed Dawson. Smart hill climbing finds better Boolean functions. In Workshop on Selected Areas in Cryptology (SAC), pages 5063, Ottawa, Canada, August 1997.