Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 5; April-June, 2015 pp. 459-463

http://www.krishisanskriti.org/acsit.html

Generation of Highly Nonlinear Boolean Function with Maximum Algebraic Degree

Renu Rawal[1] and Dheeraj Kumar Sharma[2]

[1,2]Department of Electronics & Communication Engineering, National Institute of Technology Kurukshetra, India
E-mail: [1]rawal.renu1@gmail.com, [2]sharma986dheeraj@gmail.com

Abstract—*Boolean functions play a central role in the design of information protection systems. The strength of cryptographic systems against any potential attack is determined by several criteria which the cryptographic functions should meet. Nonlinearity is a very important cryptographic criterion proposed against linear cryptanalysis. In order to linear attacks, Boolean functions used in many stream ciphers should possess high nonlinearity. To resist algebraic attacks it should have high value of algebraic degree. This paper presents main result to find highly nonlinear Boolean functions with maximum algebraic degree. A wide range of approaches have been adopted in the search of Boolean functions that excel in terms of several cryptographic criterions. In this paper, we present a new scheme based on Particle Swarm Optimization Algorithm to generate Boolean functions which satisfy high nonlinearity with maximum algebraic degree. Proposed scheme generates strong Boolean functions with desired values of these characteristics for input variables varying from 4 to 9.*

1. INTRODUCTION

An important domain of cryptography is the design and study of stream ciphers. For allowing resistance to attacks, the pseudo-random generator in such cipher must not be linear, and in fact must behave very differently from linear generators. It is often a Boolean function which ensures such non-linearity. The Boolean function must be balanced to avoid some straightforward weakness of the cipher and it must allow resistance to all known attacks, mainly the linear attack which obliges the function to have high nonlinearity, the algebraic attack which needs the function to have high algebraic degree.

Boolean functions are the building blocks of symmetric cryptographic systems. They are used for S-box design in block ciphers and utilized as nonlinear filters and combiners in stream ciphers. To resist the known attacks on each cryptosystem, Boolean functions should satisfy various criteria simultaneously. The study of the properties of the substitution transformations of DES has resulted in nonlinearity criteria for Boolean functions. Perfect nonlinear Boolean functions, also called *bent function*, are defined to be at maximum hamming distance from affine functions. Those functions of great importance in cryptography seem to be rare and very few are known. They are neither balanced nor correlation-immune and have degrre <= $\frac{n}{2}$ [1-4]. So, it seems useful to define a larger class of Boolean functions, containing maximum algebraic degree and preserving a high level of nonlinearity.

For many years, heuristic techniques have been shown to be an effective and flexible way of generating strong Boolean functions and s-boxes. Heuristic techniques involve directed search methods and have been successful in not only finding functions with "good" properties but also are able to produce a large number of such functions. Some of the best known techniques for heuristic optimization include Hill Climbing, Genetic Algorithms, Simulated Annealing and Particle Swarm Optimization. Examples of significant results produced by these methods can be found in [5], [6], [7] respectively.

Among the advantage of heuristic techniques over algebraic constructions is the ability of heuristics to generate a large number of Boolean functions with the desired properties. The non-deterministic nature of heuristic techniques results in randomness that tends to generate predominantly strong functions with a complex structure. While algebraic constructions are typically able to produce optimal functions. The basis of some algebraic constructions results in weaker functions with a structure which is more easily cryptanalysed.

In this paper we present simple PSO (Particle Swarm Optimization) method which is to be powerful tools for generating a large number of boolean functions with target cryptographic properties for enhancing security. This method provides researchers with alternative means for generating strong boolean functions for applications which require the existence of the cryptographic properties they are able to produce. This method generates a large number of boolean functions with good properties in a short period of time. This method is able to successfully produce a large number of examples of boolean functions with almost all of the optimal cryptographic property combinations: high nonlinearity and maximum algebraic degree, that have been discovered. The results obtained are particularly significant given that the algorithm used is substantially simpler than existing heuristic techniques. In addition, this method is able to efficiently

generate a wide range of functions with these optimal combinations of properties.

The remainder of this paper is organized as follows. In Section 2, some basic definitions and terms of BFs are provided while Section 3 presents the PSO algorithm. In Section 4, proposed method for generating boolean functions with maximum algebraic degree for $4 \leq N \leq 9$ is given, experimental results supporting the value of this method in generating highly nonlinear boolean functions is given in section 5. In Section 6 we draw conclusion about the method we have developed and make some suggestions for future work which could be done in this area.

2. PRELIMINARIES

We now introduce the notation used throughout this paper and present some necessary and well established definitions.

Let us denote the vector space Z_2^n is the space of all n-tuples of elements x = (x_1, x_2, x_3…, x_n) of elements from Z_2 with the standard operations. By "$\oplus$ " we denote the addition over Z_2^n for all $n \geq 1$. If x = (x_1, x_2, x_3…, x_n) and y = (y_1, y_2, y_3…, y_n) are in Z_2^n, we define the scalar (or inner) product by x.y = $x_1y_1 \oplus x_2y_2 \oplus x_ny_n$. In$Z_2^n$, let **0** and **1** denote the zeros vector and the all-one vector, respectively.

We call any function from Z_2^n to Z_2 a *Boolean function* in n-variables and denote the set of all Boolean functions by B_n. A BF is commonly represented in 'Truth Table' or 'Algebraic Normal Form' (ANF). Truth table of an n-variable BF *f(x)* is the binary output vector and contains 2^n elements each belongs to set *{0,1}*. So a Boolean function *f*(x) is also interpreted as the output column of its truth table *f*, i.e., a binary string of length 2^n. Since *f* can be expressed as a unique polynomial in n coordinates x_1, x_2, x_3…, x_n. A more convenient form of a boolean function is its polarity truth table representation, denoted by $\hat{f}(x)$, containing 2^n elements $\in \{1, -1\}$. $\hat{f}(x)$ may be derived from *f*(*x*) by the equation, $\hat{f}(x) = 1 - 2f(x)$.

The Algebraic Normal Form (ANF) is a multivariate polynomial representation of a boolean function. The variables of polynomial are the values of the input bits. The coefficients $a_i \in \{0, 1\}$ ($i = 0,..,n$) form the elements of the ANF of *f*(*x*), an *n*-variable boolean function. Every ANF representation corresponds to a unique boolean function truth table. The Algebraic Normal Form of a boolean function of *n* variables is written in the form:

$f(x) = a_0 \oplus a_1x_1 \oplus a_2x_2 \oplus \ldots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \ldots \oplus a_{(n-1)n}x_{n-1}x_n \oplus \ldots \oplus a_{123\ldots n}x_1x_2x_3\ldots x_n$.

The algebraic degree, or simply degree, of a boolean function is the order of the largest product term in the function's ANF, which we shall denote by *deg*.

A useful measure of a BF is its Hamming weight *hw(f)* which is the number of ones in its truth table. Mathematically, $hw(f) = \sum_{x=0}^{2^n-1} f(x)$. Hamming distance between two n-variable BFs, f(x) and g(x) denoted by hd(f,g) represents the number of differing corresponding elements between two truth tables and evaluated as $\mathrm{hd}(f, g) = \sum_{x=0}^{2^n-1}(f(x) \oplus g(x))$.

Correlation between two n-variable BFs, f(x) and g(x), is viewed as the degree of similarity expressed in terms of hd(f,g) and numerically represented by a real number between 0 and 1 as the correlation coefficient, $\mathrm{cc}(f, g) = 1 - \frac{\mathrm{hd}(f,g)}{2^n-1}$.

Let ω denote the bit string representation of an integer between 0 and $2^n - 1$. Then autocorrelation spectrum of f is defined as $\hat{r}_f(\omega) = \sum_{i=0}^{2^n-1}(-1)^{f(i)} . (-1)^{f(i\oplus\omega)}$. The autocorrelation spectrum is the sequence $(\hat{r}_f(0), \hat{r}_f(1), \ldots., \hat{r}_f(2^n - 1))$. The autocorrelation of f is the maximum absolute value, $\max_{i\neq 0}|\hat{r}_f(i)|$ in the autocorrelation spectrum. The Walsh-Hadamard spectrum of a Boolean function $f \in B_n$ contains information on the correlation between f and the various n-bit linear functions. That is to say, where *w* is an integer between 0 and 2^n-1, let $w_1w_2 \ldots w_n$ be the bitstring representation of *w*. Then entry *w* in the Walsh spectrum is equal to $(w) = \sum_{i=0}^{2^n-1}(-1)^{f(i)}.(-1)^{w.i}$.

For all *w*, *F(w)* is a real number in the range [-2^n, 2^n]. The vector representing WHT of a function is referred as Walsh Hadamard Spectrum (WHS).A balanced function is a Boolean function with an equal number of 1s and 0s in its truth table. That is an n-variable BF, *f*(x), is said to be a balanced function if hw(*f*)= $\frac{2^n}{2} = 2^{n-1}$. For a balanced function *f*, hw(*f*) = hw(*f* $\oplus$1) . In terms of its WHT, a BF is balanced if F(0) = 0. The algebraic degree of an n variable balanced BF cannot exceed n-1.

The resistance to known attacks can be quantified through some fundamental characteristics of BFs used and the design of these functions needs to consider various characteristics simultaneously. The important characteristics of BFs are discussed in following paragraphs.

2.1 Non Linearity

Nonlinearity of n-variable BF *f* represents the dissimilarity between *f* and n-variable affine function, *a*. hamming distance between *f* and *a* represents the bitwise similarity between *f* and affine function. As all affine functions are known to be cryptographically weak, more dissimilar *f* to *a*, higher its nonlinearity which resists cryptanalytic attacks those based on linear approximations. A BF with low nonlinearity value means that the function is very close to an affine function and therefore there is a reasonable probability to approximate the function by that affine function. It improves the likelihood of successful linear cryptanalysis. Thus, a BF which is highly nonlinear is cryptographically desirable as it provides stronger resistance to linear cryptanalysis. The nonlinearity of an n-variable BF *f*, *nl(f)*, is the minimum distance to the set of all n-variable affine functions. For any Boolean function *f*, nonlinearity is defined as $nl(f) =$

$\min_a\{\mathrm{hd}(f,a)\}$.Nonlinearity of the function can be evaluated Walsh hadamard transform as $l(f) = 2^{m-1} - \frac{\max |F(w)|}{2}$, where *w* is an integer between 0 and 2^n-1.

Obviously, the nonlinearity of an affine Boolean function is zero; the maximal value of the nonlinearity is given by $nl(f) \leq 2^{m-1} - 2^{\frac{m}{2}-1}$. BFs having the maximum nonlinearity are known as bent functions. A bent function exists only for even number of input variables and unbalanced as the nonlinearity of balanced Boolean functions is below the maximal value. The following bounds on nonlinearity are found in [9]

$$nl(f) \leq \begin{cases} 2^{m-1} - 2^{\frac{m}{2}-1} - 2, m\ even \\ \left\lfloor 2^{m-1} - 2^{\frac{m}{2}-1} \right\rfloor, m\ odd \end{cases}$$

where $\lfloor x \rfloor$ denotes the maximum even integer less than or equal to *x*.

In order to increase the nonlinearity of a BF, maximum absolute value of WHT (*WHTmax)* must be decreased. A function *f(x)* is uncorrelated with linear function *a(x)* when *F(a)=0*. Cryptographically, it is desirable to find BFs, which have all WHT values equal to zero. However such functions do not exist. Since sum of the squares of WHT values is same constant, 2^{2^m} for every BF [13], so there is an opportunity to minimize affine correlation and maximize nonlinearity of functions.

3. PARTICLE SWARM OPTIMIZATION ALGORITHM

The Particle Swarm Optimization algorithm (abbreviated as PSO) is a novel population-based stochastic search algorithm and an alternative solution to the complex non-linear optimization problem. The PSO algorithm was first introduced by Dr. Kennedy and Dr. Eberhart in 1995 and its basic idea was originally inspired by simulation of the social behavior of animals such as bird flocking, fish schooling and so on. It is based on the natural process of group communication to share individual knowledge when a group of birds or insects search food or migrate and so forth in a searching space, although all birds or insects do not know where the best position is. But from the nature of the social behavior, if any member can find out a desirable path to go, the rest of the members will follow quickly [10].

In PSO, each member of the population is called a particle and the population is called a swarm. Starting with a randomly initialized population and moving in randomly chosen directions, each particle goes through the searching space and remembers the best previous positions of itself and its neighbors. Particles of a swarm communicate good positions to each other as well as dynamically adjust their own position and velocity derived from the best position of all particles. The next step begins when all particles have been moved. Finally, all particles tend to fly towards better and better positions over the searching process until the swarm move to close to an optimum of the fitness function [11].

The Particle Swarm Optimization (PSO) algorithm is a multi-agent parallel search technique which maintains a swarm of particles and each particle represents a potential solution in the swarm. All particles fly through a multidimensional search space where each particle is adjusting its position according to its own experience and that of neighbors. Suppose x_i^t denote the position vector of particle in the multidimensional search space at time step *t*, then the position of each particle is updated in the search space by

$$x_i^{t+1} = x_i^t + v_i^{t+1} \text{ with } x_i^0 \sim U(x_{min}, x_{max}) \quad (1)$$

Where, v_i^t is the velocity vector of particle that drives the optimization process and reflects both the own experience knowledge and the social experience knowledge from the all particles; $\mathrm{U}(x_{min}, x_{max})$ is the uniform distribution and x_{min} and x_{max} are its minimum and maximum values respectively.

Therefore, in a PSO method, all particles are initiated randomly and evaluated to compute fitness together with finding the personal best (best value of each particle) and global best (best value of particle in the entire swarm). After that a loop starts to find an optimum solution. In the loop, first the particles' velocity is updated by the personal and global bests, and then each particle's position is updated by the current velocity. The loop is ended with a stopping criterion predetermined in advance.

As we want to find the optimal BFs having different desired cryptographic properties such as nonlinearity, algebraic immunity, algebraic degree and correlation immunity where a Boolean function *f* is a mapping from Z_n^2 into Z_2. So I want an optimization technique which provides good optimal solutions for discrete valued input. In 1997, Kennedy and Eberhart firstly extended the basic PSO algorithm to the discrete space to solve this problem [12]. They developed the PSO to operate on the binary search spaces, because real-valued domains can be transformed into the binary-valued domains. The proposed algorithm is called binary PSO (BPSO) algorithm [11]. In BPSO, a particle's velocity is connected to the possibility that the particle's position takes a value of 0 or 1. The update equation for the velocity of the particles is given by

$$v_{ij}^{t+1} = v_{ij}^t + c_1 r_{1j}^t \left(P_{best,i}^t - x_{ij}^t \right) + c_2 r_{2j}^t \left(G_{best} - x_{ij}^t \right) \quad (2)$$

Now, the j^{th} bit of the i^{th} particle, x_{ij}^t is updated by

$$x_{ij}^t = \begin{cases} 1\ if\ u_{ij}^t < s_{ij}^t \\ 0\ if\ u_{ij}^t \geq s_{ij}^t \end{cases} \quad (3)$$

where, u_{ij}^t is a random number selected from a uniform distribution in (0, 1), and s_{ij}^t is the sigmoid function , denoted by,

$$s_{ij}^{t} = \frac{1}{1+e^{-v_{ij}^{t+1}}} \quad (4)$$

4. PSO BASED BOOLEAN FUNCTION GENERATION

Algebraic Degree and Non Linearity are considered as most desirable characteristics of BFs to be generated. In this technique we get an optimal solution either to maximize the fitness function or to minimize the cost function and also there is flexibility in choosing values of desired characteristics We define a cost function to get an Boolean function having optimum values of the above cryptographic properties as follows[7]:

$$\text{cost}(f) = (2^{n-1} - \text{NL}) - (\text{algebraic degree})$$

where NL indicates the non-linearity of the BF.

Steps involved in the algorithm are as follows:

i. Initialize max. no. of particles (population size), maximum number of iterations, initial velocities and acceleration coefficients c_1 & c_2. Set desired values of required characteristics, i.e. Algebraic Degree and Non Linearity for BF to be generated.

ii. Randomly initialize the population of BFs. Represent each BF by its truth table.

iii. Evaluate the values of Algebraic Degree and Non Linearity of BFs.

iv. Compute value of cost using above formula for each particle i.e. for each BF.

v. Consider these values of cost function as the personal best solution and the minimum value of cost function is considered as global best solution and corresponding particles (BFs) are known as personal best position and global best position respectively.

vi. Update the velocity and position of each particle using equation (2) and equation (3) given in section 3.

vii. Now compute value of cost for each updated position i.e. particle (BF) and also find the minimum cost.

viii. Update the personal best solution and global best solution by comparing these updated solutions with previous solutions.

ix. Stop the process when cost function value is pre -defined threshold value and corresponding BF will be the desired output. Otherwise repeat the process (vi) to (ix) for new population.

x. Terminate process after pre-defined number of iterations.

5. SIMULATION RESULTS AND OBSERVATIONS

After applying the PSO method given in Sect. 4 using Matlab software for search of Boolean function having optimum value of nonlinearity with high algebraic degree, we get the desired Boolean functions. Applying our method, we have constructed such functions of 4 to 9 variables as shown in Table 1. The parameters taken to construct functions are listed in Table 2.

Table 1: Simulation results of proposed algorithm.

No. of input variables (n)	No. of iterations (N)	Algebraic Degree (AD)	Non-linearity (NL)		
			Maximum possible	Genetic algorithm [8]	Proposed algorithm
4	21	2	6	4	6
5	20	3	12	12	12
6	500	5	28	16	26
7	500	6	56	38	53
8	500	8	120	86	111
9	500	9	240	196	227

Table 2: Parameters used for simulation

Parameter name	Parameter values
Population size	20
Maximum no. of iterations	500
Initial velocity	0
Acceleration coefficients	1

6. CONCLUSION AND FUTURE SCOPE

In this piece of work, we have developed a new method to construct desired Boolean functions. By applying our method, we got at least as optimum results with highly nonlinear function. Proposed scheme give these results within small number of iterations as well as population size. Incorporation of such generated strong Boolean functions in the design of crypto algorithms protects such developed cryptosystems from various attacks such as algebraic, linear, differential, statistical attacks etc. and strengthens their security to safeguard communications in various information security applications.

We can further generate stronger Boolean functions by either increasing the population size or by increasing the number of iterations or both as well.

REFERENCES

[1] Palash Sarkar and Subhamoy Maitra, "Construction of Nonlinear Boolean Functions with Important Cryptographic Properties", *advances in cryptology- eurocrypt' 2000, Springer-Verlag*, May 2000.

[2] Palash Sarkar and Subhamoy Maitra, "Efficient Implementation of Cryptographically Useful "Large" Boolean Functions", *IEEE transactions on COMPUTERS*, VOL. 52,in April 2003.

[3] Wei-Guo Zhang, Member, IEEE, and Enes Pasalic, "Generalized Maiorana–McFarland Construction of Resilient Boolean Functions with High Nonlinearity and Good Algebraic Properties.", *IEEE transactions on INFORMATION THEORY*, VOL. 60, in October 2014.

[4] Rajni Goyal and Shiv Prasad Yadav, "An evolutionary approach to construct cryptographically strong Boolean functions",

International Journal of System Assurance Engineering & Management, Springer-Verlag, in April 2012.

[5] William Millan, Andrew Clark, and Ed Dawson, "Smart hill climbing finds better boolean functions", In *Selected Areas in Cryptology (SAC'97)*, pages 50–63, August 1997.

[6] William Millan, Andrew Clark, and Ed Dawson, "Heuristic design of cryptographically strong balanced boolean functions", In *Advances in Cryptology - EUROCRYPT'98*, volume 1403, Springer Verlag, 1998.

[7] James McLaughlin and John A. Clark, "Evolving balanced Boolean functions with optimal resistance to algebraic and fast algebraic attacks, maximal algebraic degree, and very high nonlinearity", Cryptology eprint In *International Association for Cryptologic Research*(IACR) in 2013.

[8] Rajesh Asthana, Neelam Verma and Ram Ratan (Defence Research and Development Organization, Scientific Analysis Group), "Generation of Boolean Functions Using Genetic Algorithm for Cryptographic Applications", *IEEE International Advance Computing Conference (IACC)* in September 2014.

[9] Sihem Mesnager, " Improving the Lower Bound on the Higher Order Nonlinearity of Boolean Functions with Prescribed Algebraic Immunity", *IEEE TRANSACTIONS ON INFORMATION THEORY*, VOL. 54, AUGUST 2008, pp. 3656-3662.

[10] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization", *IEEE Swarm Intelligence Symposium*, 2007.

[11] Frans van den Bergh and Andries P. Engelbrecht, Member, IEEE, "A Cooperative Approach to Particle Swarm Optimization", *IEEE transactions on Evolutionary Computation*, vol. 8, in June 2004.

[12] Mohammad Teshnehlab and Mahdi Aliyari Shoorehdeli Mojtaba Ahmadieh Khanesar, "A Novel Binary Particle Swarm Optimization", in Proceedings of the 15th *Mediterranean Conference on Control and Automation*, Greece, July 2007.

[13] J. A. Clark, J. L. Jacob, S. Stepney, S. Maitra and W. Millan, "Evolving Boolean functions satisfying multiple criteria", *INDOCRYPT'02, Springer-Verlag*, 2002.