Paper

A study of differences between bent functions constructed using Rothaus method and randomly generated bent functions

Anna Grocholewska-Czuryło

Abstract — Bent functions, having the highest possible nonlinearity, are among the best candidates for construction of *S*-boxes. One problem with bent functions is the fact that they are hard to find among randomly generated set of Boolean functions already for 6 argument functions. There exist some algorithms that allow for easy generation of bent functions. The major drawback of these algorithms is the fact that they rely on deterministic dependencies and are only able to generate bent functions belonging to one specific class. In our paper we present an efficient generator of random bent functions of more than 4 arguments. Resulting functions are not bounded by constraints described above. The generator operates in algebraic normal form domain (ANF). We also present our result on comparing the performance of *S*-boxes build using our bent function generator versus a standard method of bent function construction. We also give some directions for further research.

Keywords — block ciphers, S-boxes, bent functions, construction, random generation, nonlinearity.

1. Introduction

In block ciphers based on *S*-boxes, the cryptographic strength (i.e. resistance to cryptanalysis attack) depends on the nonlinear properties of an *S*-boxes used to build the cipher. *S*-boxes are built from Boolean functions, so quality of each and every function constituting an *S*-box is of a greatest importance. Another major consideration for *S*-box construction is the way functions that form an *S*-box "interact" (behave as a group of functions) – what are the cryptographic properties of an *S*-box as a whole. These are two main factors that affect cipher's cryptographic performance.

The properties of Boolean functions have been extensively studied. The quality of a single, cryptographically strong Boolean function is measured by its cryptographic properties. The criteria against which a function quality is measured are mainly nonlinearity, balancedness, avalanche and propagation criteria.

The qualities that single Boolean functions should posses to be good candidates for *S*-box construction are very similar to those that should be characterizing a good (strong) *S*-box (taken as a linear combination of constituting functions). The nonlinear properties are by far the most important. In recent years a class of highly nonlinear functions attracted a lot of researchers' attention – a class of bent Boolean functions. These functions have in fact the highest possible nonlinearity (they also have very good propagation characteristics and are nearly balanced – another important criterion for good cryptographic function – special algorithm have been proposed by different authors for transforming bent functions into balanced Boolean functions while maintaining high level of nonlinearity).

However one major drawback is the fact that bent functions are not easily constructed (i.e. their constructions are time consuming). Trying to find bent functions by pure random search is also virtually impossible (already for 6-argument functions, only one in $2.9 \cdot 10^{-10}$ Boolean functions is bent). Also, an *S*-box that is constructed using only bent functions does not necessarily possess the best nonlinear qualities. So usually a number of different *S*-boxes should be generated and tested, and then only the best *S*-boxes should be selected for incorporating in block cipher. This approach demands fast *S*-box generation which in turn translates to fast bent function generation.

The remainder concentrates on efficient random bent function generation and using such generated function for *S*-box construction.

2. Preliminaries

We use square brackets to denote vectors like $[a_1,\dots,a_n]$ and round brackets to denote functions like $f(x_1,\dots,x_n)$.

Boolean function. Let GF(2) $=<\sum,\oplus,\bullet>$ be two-element Galois field, where $\sum=\{0,1\}$, $\oplus$ and $\bullet$ denotes the sum and multiplication mod 2, respectively. A function $f:\sum^n\rightarrow\sum$ is an *n*-argument Boolean function. Let $z=x_1\cdot 2^{n-1}+x_2\cdot 2^{n-2}+\ldots+x_n\cdot 2^0$ be the decimal representation of arguments $(x_1,x_2,\dots,x_n)$ of the function f. Let us denote $f(x_1,x_2,\dots,x_n)$ as y_z. Then $[y_0,y_1,\dots,y_{2^n-1}]$ is called a truth table of the function f.

Linear and nonlinear Boolean functions. An *n*-argument Boolean function f is linear if it can be represented in the following form: $f(x_1,x_2,\dots,x_n)=a_1x_1\oplus a_2x_2\oplus\ldots a_nx_n$.

Let L_n be a set of all n-argument linear Boolean functions. Let $M_n = \{g : \sum^n \to \sum \mid g(x_1, x_2, \ldots, x_n) = 1 \oplus f(x_1, x_2, \ldots, x_n)$ and $f \in L_n\}$. A set $A_n = L_n \cup M_n$ is called a set of n-argument affine Boolean functions. A Boolean function $f : \sum^n \to \sum$ that is not affine is called a nonlinear Boolean function.

Balance. Let $N_0[y_0, y_1, \ldots, y_{2^n-1}]$ be a number of zeros (0's) in the truth table $[y_0, y_1, \ldots, y_{2^n-1}]$ of function f, and $N_1[y_0, y_1, \ldots, y_{2^n-1}]$ be number of ones (1's). A Boolean function is balanced if $N_0[y_0, y_1, \ldots, y_{2^n-1}] = N_1[y_0, y_1, \ldots, y_{2^n-1}]$.

Algebraic normal form. A Boolean function can also be represented as a maximum of 2^n coefficients of the algebraic normal form. These coefficients provide a formula for the evaluation of the function for any given input $x = [x_1, x_2, \ldots, x_n]$:

$$f(x) = a_0 \oplus \sum_{i=1}^{n} a_i x_i \oplus \sum_{1 \le i < j \le n} a_{ij} x_i x_j \oplus \ldots \oplus a_{12\ldots n} x_1 x_2 \ldots x_n,$$

where $\sum, \oplus$ denote the modulo 2 summation.

The order of nonlinearity of a Boolean function $f(x)$ is a maximum number of variables in a product term with non-zero coefficient a_J, where J is a subset of $\{1, 2, 3, \ldots, n\}$. In the case where J is an empty set the coefficient is denoted as a_0 and is called a zero order coefficient. Coefficients of order 1 are $a_1, a_2, \ldots, a_n$, coefficients of order 2 are $a_{12}, a_{13}, \ldots, a_{(n-1)n}$, coefficient of order n is $a_{12\ldots n}$. The number of all ANF coefficients equals 2^n.

Let us denote the number of all (zero and non-zero) coefficients of order i of function f as $\sigma_i(f)$. For n-argument function f there are as many coefficients of a given order as there are i-element combinations in n-element set, i.e. $\sigma_i(f) = \binom{n}{i}$.

Hamming distance. Hamming weight of a binary vector $x \in \sum^n$, denoted as $hwt(x)$, is the number of ones in that vector.

Hamming distance between two Boolean functions $f, g : \sum^n \to \sum$ is denoted by $d(f, g)$ and is defined as follows:

$$d(f, g) = \sum_{x \in \sum^n} f(x) \oplus g(x).$$

The distance of a Boolean function f from a set of n-argument Boolean functions X_n is defined as follows:

$$\delta(f) = \min_{g \in X_n} d(f, g),$$

where $d(f, g)$ is the Hamming distance between functions f and g. The distance of a function f from a set of affine functions A_n is the distance of function f from the nearest function $g \in A_n$.

The distance of function f from a set of all affine functions is called the nonlinearity of function f and is denoted by N_f.

Bent functions. A Boolean function $f : \sum^n \to \sum$ is perfectly nonlinear if and only if $f(x) \oplus f(x \oplus \alpha)$ is balanced for any $\alpha \in \sum^n$ such that $1 \le hwt(\alpha) \le n$.

For a perfectly nonlinear Boolean function, any change of inputs causes the change of the output with probability of 1/2.

Meier and Staffelbach [16] proved that the set of perfectly nonlinear Boolean functions is the same as the set of Boolean bent functions defined by Rothaus [5].

Perfectly nonlinear functions (or bent functions) have the same, and the maximum possible distance to all affine functions. So their correlation to any affine function is consistently bad (minimal). Linear cryptanalysis works if it is possible to find a good linear approximation of the S-box.

Bent functions are not balanced. This property prohibits their direct application in S-box construction, however there exist numerous methods for modifying bent function in such a way so that the resulting function is balanced and still maintains the good cryptographic properties of a bent function [16]. Hamming weight of a bent function equals $2^{n-1} \pm 2^{n/2-1}$.

Differential analysis [18] can be seen as an extension of the ideas of attacks based on the presence of linear structures [3]. As perfect nonlinear Boolean function have maximum distance to the class of linear structures (equal to 2^{n-2}), they are a useful class of functions for constructing mappings that are resistant to differential attacks.

Bent functions exist only for even n. The nonlinear order of bent functions is bounded from above by $n/2$ for $n > 2$. The number of Boolean bent function for $n > 6$ remains an open problem.

3. Constructing bent functions

There exist a number of algorithms for constructing bent functions. As an example let's consider the following [8, 12].

Method 1. Let B_n denote a set of bent functions $f : \sum^n \to \sum$ with n even. Given a set of bent functions B_6, bent functions in B_8 can be constructed using the following method (Method 1).

Let $a, b \in B_6$. Then the function $f : \sum^8 \to \sum$ defined by:

$$f(x_0 \ldots, x_7) = \begin{cases} a(x_0 \ldots x_5), & x_6 = 0,\ x_7 = 0 \\ a(x_0 \ldots x_5), & x_6 = 0,\ x_7 = 1 \\ b(x_0 \ldots x_5), & x_6 = 0,\ x_7 = 0 \\ b(x_0 \ldots x_5), \oplus 1, & x_6 = 0,\ x_7 = 0 \end{cases}$$

is bent [8]. Rearrangements of the 64 blocks in the expression above also result in bent functions.

Another method for bent function construction was given by Rothaus in [5].

Method 2. Let $x = (x_1, \ldots, x_n)$ and let $a(x)$, $b(x)$ and $c(x)$ be bent functions such that $a(x) \oplus b(x) \oplus c(x)$ is also bent. Then a function $f(x, x_{n+1}, x_{n+2}) = a(x)b(x) \oplus b(x)c(x) \oplus c(x)a(x) \oplus [a(x) \oplus b(x)]x_{n+1} \oplus [a(x) \oplus c(x)]x_{n+2} \oplus x_{n+1}x_{n+2}$ is bent.

Most of the known bent function constructions take bent functions of n arguments as their input and generate bent functions of $n+2$ arguments. One major drawback of these methods is the fact that they are deterministic. Only short bent functions ($n = 4$ or 6) are selected at random and the resulting function is obtained using the same, deterministic formula every time. Even if there is some "random" element in such generation (like adding a linear term to the resulting bent function) it does not bring any new quality to the generated function.

4. Generating bent functions

To overcome some of the limitations and possible weaknesses of the bent functions construction methods described above a new algorithm of random bent functions generation have been proposed [24].

As already mentioned earlier, drawing bent functions at random is not feasible already for small number of arguments ($n > 6$). To make such generation possible, an algorithm was designed that generates random Boolean functions in algebraic normal form thus making use of some basic properties of bent functions to considerably narrow the search space. This makes the generation of bent functions feasible for $n \geq 8$ even on a standard PC machine. The algorithm for the generation of bent functions in ANF domain takes as its input the minimum and maximum number of ANF coefficients of every order that the resulting functions are allowed to have. Since the nonlinear order of bent functions is less or equal to $n/2$, clearly in ANF of a bent function cannot be any ANF coefficient of order higher then $n/2$. This restriction is the major reason for random generation feasibility, since it considerably reduces the possible search space.

The number of ANF coefficients of orders less or equal to $n/2$ can be fixed or randomly selected within allowed range (i.e. between 0 and $\sigma(f) = \binom{n}{i}$ for order i). If the number of coefficients for a given order i is fixed then all generated functions will have the same number of coefficients of that order, but the coefficients themselves will be different in each generated function. If the number of coefficients for a given order i is randomly selected then all generated functions will not only have different coefficients but also the number of coefficients of order i will vary from function to function. It is of course possible to fix the number of coefficients for some orders and have varied number of coefficients for other orders.

One important consequence of this approach is the possibility of prohibiting the generation of bent functions which are merely a linear transformations of other bent functions. This is easily achieved by setting the number of coefficients of order 0 and 1 to 0. So in the ANF of the resulting functions there will be no linear part. Bent functions of any order can be generated with this method, simply by setting any higher order coefficients to 0. Homogenous bent functions can also be generated easily.

One drawback of the method is the fact that it does not guarantee the generation of bent functions without repetitions, although the chance of generating two identical bent functions is minimal with any reasonably selected ranges of number of ANF coefficients. However, if avoiding repetitions is an absolute requirement, the set of generated bent functions must be checked for duplicates.

The limitations of this approach are twofold. First there is a feasibility limit. Number of possible functions grows with the number of coefficients of higher orders ($i > 2$) and generating a bent function quickly becomes infeasible. So the algorithm works best with the low number of higher order coefficients (e.g. < 6 for $n = 8$ and order $i = 3$ and 4). Due to the above limitation, this method does not generate all possible bent functions with equal probability. In principle, it would be possible but is not feasible for the reason described above. One has to limit the number of higher order coefficients and at the same time prohibit the generation of some bent functions.

5. Comparing pairs of bent functions

In this section some comparative results are presented. Three sets of 8 argument bent Boolean functions are analyzed: bent functions constructed using Method 1, bent functions constructed using method given in [22] (Maiorana functions with permuted inputs) and randomly generated bent functions. For random, distinct i, j the nonlinearity of $f_i \oplus f_j$ was calculated. Figures 1 and 2 show the resulting nonlinearity distribution (in percentage).

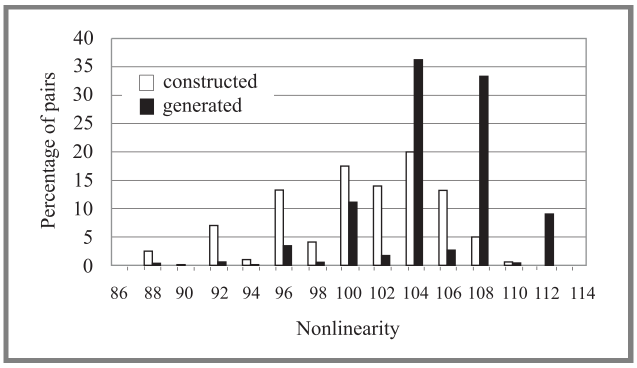

Fig. 1. Pairs nonlinearity distribution. Constructed bent (Method 1) versus generated bent.

The random bent functions were generated with the following parameters: number of 2nd order coefficients was between 7 and 14 (statistically that yields the highest number of bent functions), number of 3rd order coefficients was fixed at 2 and number of 4th order ANF coefficients was also fixed at 2. There were no coefficients of order 0 and 1 to prevent the occurrences of bent functions that would be just a linear transformations of one another.

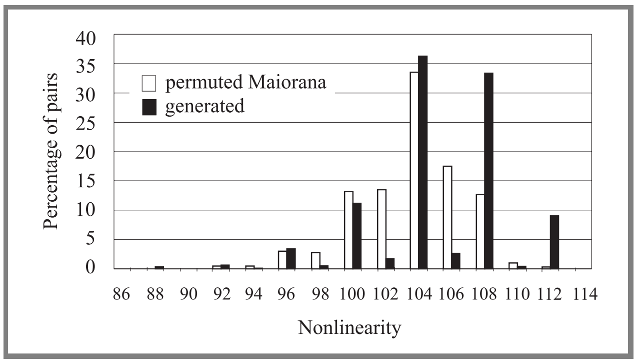

Fig. 2. Pairs nonlinearity distribution. Constructed bent (permuted Maiorana) versus generated bent.

For randomly generated functions the distribution is shifted towards higher values (i.e. pairs have better nonlinearity) and is also much more narrow - more appropriate pairs can be found in this set of functions. The results obtained in our experiments are also better then those presented in [22] for a special subsets of bent functions: Maiorana functions with permuted inputs.

6. Comparing S-boxes

In this section we present some properties of S-boxes build using randomly generated bent function. We give comparative results of the performance of S-boxes build of bent functions constructed using a method introduced by Rothaus [5] (Method 2 from Section 3), bent functions generated with our algorithm described in this paper and random Boolean functions (not bent). Two properties of S-boxes were measured: feasibility of a linear approximation which is a measure of S-box resistance against linear cryptanalysis and nonlinearity of the S-box which is one of the major criteria of cryptographic quality.

One has to note that for “real-life” applications bent functions would have to be modified to be balanced prior to their use in S-boxes. Such modification algorithms are beyond the scope of this paper, where main focus is kept on bent functions.

6.1. Linear approximations of S-boxes

We used a method of linear approximation as described in [23].

By linear approximation of a Boolean function $h : \sum^n \to \sum^m$, written as $Y = h(X)$, we mean any equation of the form:

$$\sum_{i\in Y'} y_i = \sum_{j\in X'} x_j, \text{ for } Y' \subseteq \{1,2,\dots,m\}, X' \subseteq \{1,2,\dots,n\},$$

fulfilled with the probability of $p = N(X',Y')/2^n$, where $N(X',Y')$ denotes the number of pairs (X,Y) fulfilling the equation, and $\sum$ is a modulo 2 summation. The sets if indices X',Y' are called input and output masks.

The measure of linear approximation effectiveness is the value of a probability $\Delta p = |p - 1/2|$ called differential probability. For a fixed n a measure of effectiveness can also be defined as a value of $\Delta N(X',Y') = = |N(X',Y') - 2^{n-1}|$.

In our experiment we tested linear approximations of 6 x 6 S-boxes, i.e. functions $Y = h(X) : \sum^6 \to \sum^6$, where sub-functions of function h were constructed bent functions, randomly generated bent functions and random functions (Fig. 3). The distribution of the best approximations was tested, i.e. maximum value of $\Delta N(X',Y')$ among all possible sets of input and output masks (except empty output mask). For each type of functions 10 000 of random S-boxes were tested. The number of S-boxes is given on Y (value) axis. The X (category) axis gives the values of the best approximations (higher value means better approximation so worse S-box).

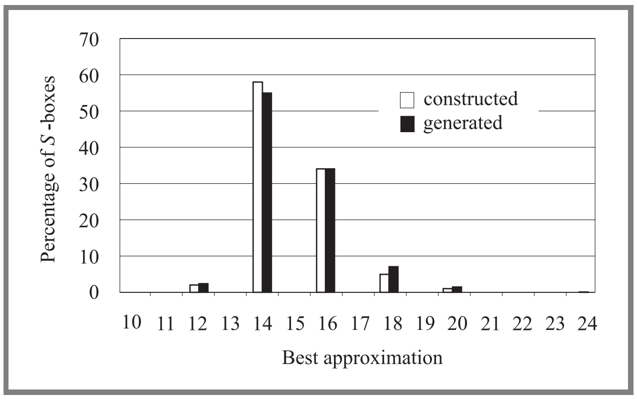

Fig. 3. Best S-box approximation distribution. Constructed (Rothaus) versus generated bent.

Differences between S-boxes build of bent functions constructed using Rothaus method (Method 2) and S-boxes build from randomly generated bent functions are not very evident.

6.2. Nonlinearity

Now we will show the results of testing the S-boxes for high nonlinearity (Fig. 4). We consider 6 x 8 S-boxes (each S-box is constructed of six 8-argument functions).

The nonlinearity of an S-box, so a function $F : \sum^n \to \sum^m$ such that $F(x) = \big(f_1(x), f_2(x), \dots, f_m(x)\big)$ i $x \in \sum^n$, is calculated as minimal nonlinearity of all linear combinations of F’s sub-functions. The nonlinearity of a S-box is then defined as follows:

$$N_F = \min\{N_{f_J} \mid f_J = \sum_{i\in J} f_i,\ J \subseteq (1,2,\dots,m)\}.$$

To calculate a nonlinearity of a single S-box 2^m linear combinations have to be constructed and their distance to affine functions calculated. The lowest of all calculated nonlinearities (distances to affine functions) is the nonlinearity of the S-box.

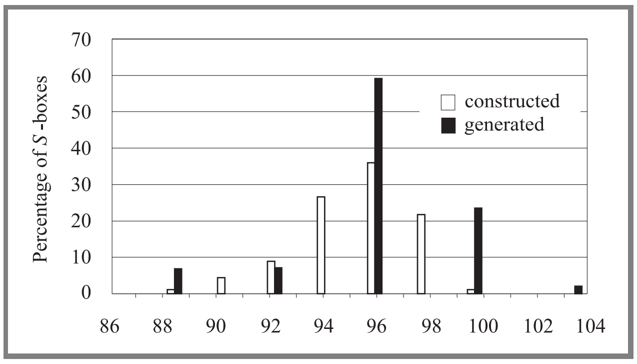

Fig. 4. *S*-box nonlinearity distribution. Constructed (Rothaus) versus generated bent.

Among *S*-boxes built from generated bent functions there exist *S*-boxes of the highest found nonlinearity of 104. There is also about 20 times more *S*-boxes of very high nonlinearity of 100 then in the case of *S*-boxes build from constructed bent functions or random balanced functions. This means that using randomly generated bent functions may lead to constructing *S*-boxes of better cryptographic qualities in less time.

However, one has to note the fact that in case of randomly generated bent functions there are also *S*-boxes of relatively poor nonlinearity (like 80). So building *S*-boxes from these functions requires (more then in other cases) careful checking the resulting *S*-boxes for possibly low nonlinearity.

7. Conclusions

From the results presented in this paper it seems that random generated bent functions offer an interesting alternative to construction methods. Not only nonlinear characteristics of these functions are equal or better then those of constructed bent functions but also generated functions have a very compact (small) algebraic normal form which can be utilized for efficient storage and fast cryptographic routines.

Next step in randomly generated bent function assessment will be checking their avalanche and propagation criteria, also when incorporated into *S*-boxes.

Perhaps a combined method of ANF generation of relatively short bent functions (i.e. $n \leq 10$) and then supplying them as an input for deterministic construction can yield some interesting results. Such functions would also have to be tested to verify their possibly superior cryptographic qualities.

References

[1] L. J. O'Connor, "An analysis of a class of algorithms for *S*-box construction", *J. Cryptol.*, vol. 7, no. 3, pp. 133–152, 1994.

[2] C. E. Shannon, "Communication theory of secrecy systems", *Bell Syst. Techn. J.*, vol. 28, pp. 656–715, 1949.

[3] K. Nyberg, "Perfect nonlinear *S*-boxes", in *Advances of Cryptology – EUROCRYPT'91, LNCS*. Springer, 1991, vol. 547, pp. 378–386.

[4] J. Seberry, X. M. Zhang, and Y. Zheng, "Systematic generation of cryptographically robust *S*-boxes", in *Proc. 1st ACM Conf. Comput. Commun. Secur.*, 1993.

[5] O. S. Rothaus, "On bent functions", *J. Combinat. Theory*, vol. 20, pp. 300–305, 1976.

[6] B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle, "Propagation characteristics of Boolean functions", in *Advances in Cryptology – EUROCRYPT'90, LNCS*. Springer, 1991, vol. 473, pp. 161–173.

[7] J. F. Dillon, "A survey of bent functions", *NSA Techn. J.*, special issue, pp. 191–215, 1972.

[8] C. M. Adams and S. E. Tavares, "Generating and counting binary bent sequences", *IEEE Trans. Inform. Theory*, vol. IT-36, pp. 1170–1173, 1990.

[9] J. A. Maiorana, "A class of bent functions", R41, 1971.

[10] C. M. Adams, "A formal and practical design procedure for substitution permutation network cryptosystems". Ph.D. thesis, Department of Electrical Engineering, Queen's University, 1990.

[11] M. Dawson and S. E. Tavares, "An expanded set of *S*-box design criteria based on information theory and its relation to differential-like attacks", in *Advances in Cryptology – EUROCRYPT'91, LNCS*. Springer, 1991, vol. 547, pp. 352–367.

[12] J. B. Kam and G. Davida, "Structured design of substitution-permutation encryption networks", *IEEE Trans. Comput.*, vol. C-28, pp. 747–753, 1979.

[13] L. O'Connor, "An analysis of product ciphers based on the properties of Boolean functions". Ph.D. thesis, Department of Computer Science, University of Waterloo, 1992.

[14] A. F. Webster and S. E. Tavares, "On the design of *S*-boxes", in *Advances in Cryptology – CRYPTO'85, LNCS*. Springer, 1986, pp. 523–534.

[15] R. Forré, "The strict avalanche criterion: spectral properties of Boolean functions with high nonlinearity", in *Advances in Cryptology – CRYPTO'88, LNCS*. Springer, 1990.

[16] W. Meier and O. Staffelbach, "Nonlinearity criteria for cryptographic functions", in *Advances in Cryptology – EUROCRYPT '89, LNCS*. Springer, 1990, vol. 434, pp. 549–562.

[17] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[18] M. Matsui, "Linear cryptanalysis method for DES cipher", in *Proc. EUROCRYPT'93* (abstracts), 1993.

[19] R. Yarlagadda and J. E. Hershey, "A note on the eigenvectors of Hadamard matrices of order 2^n", *Linear Algebra & Appl.*, vol. 45, pp. 43–53, 1982.

[20] R. Yarlagadda and J. E. Hershey, "Analysis and synthesis of bent sequences", *Proc. IEE*, vol. 136, pp. 112–123, 1989.

[21] K. Nyberg, "Constructions of bent functions and difference sets", in *Advances in Cryptology – EUROCRYPT'90, LNCS*. Springer, 1991, vol. 473.

[22] S. Mister and C. Adams, "Practical *S*-box design", in *Workshop on Selected Areas in Cryptography (SAC '96) Workshop Record*, Queens University, 1996, pp. 61–76.

[23] K. Chmiel, "Liniowa aproksymacja funkcji *S*-bloków". Raport nr 475. Politechnika Poznańska, Katedra Automatyki, Robotyki i Informatyki, Poznań, 2000 (in Polish).

[24] R. Wicik, "Wykorzystanie szyfrów blokowych opartych o sieci podstawieniowo-przestawieniowe o dużych *S*-boksach w specjalnych sieciach telekomunikacyjnych". Rozprawa doktorska, Wojskowa Akademia Techniczna, Warszawa, 1999 (Ph.D. thesis in Polish).

[25] A. Grocholewska-Czuryło and J. Stokłosa, "Generating bent functions", in *Proc. ACS 2001*.

Anna Grocholewska-Czuryło was born in Poznań, Poland, in 1969. In 1992 she graduated with the M.Sc. degree in computer science from the Faculty of Electrical Engineering at the Poznań University of Technology. She had been working for a year in University's Super Computer Center before she has moved on to become a teacher and a research assistant at the Laboratory of Information Technology Security. She has studied and published papers on a range of topics like natural language processing, cellular automata, neural networks and has finally focused on cryptography, and *S*-box design in particular. She has earned her Ph.D. degree in 2001.
e-mail: czurylo@sk-kari.put.poznan.pl
Laboratory of Information Technology Security
Poznań University of Technology
Marii Skłodowskiej-Curie st 5
61-542 Poznań, Poland