Constructing Bent Functions for Cryptography with Binary Functions

Milos Radmanovic
Faculty of Electronic Engineering
Aleksandra Medvedeva 14, 18000 Nis, Serbia
{milos.radmanovic@elfak.ni.ac.rs}

ABSTRACT: *We in this paper deal with Bent functions which are essential for cryptographic binary functions. These functions make very small subset of the total number of bent functions, especially for large number of variables. We do not find any formal method for construction of bent functions. Hence the methods for construction of bent functions are obtained by using the discovery of bent functions within a group of functions. For experimenting the bentness across a group of functions, even for small numbers of variables, requires a lot of processing time. We in this work introduced an efficient method for bentness testing of binary functions using statistical analysis of binary functions. The standard method for bentness testing is based on the usage of the fast Walsh transform calculations. This method uses conditional testing of the initial set of particular Walsh coefficients. In the current work, the selection of coefficients for conditional testing is determined using statistical analysis of binary functions.*

Keywords: Cryptography, Binary Function, Bent, Walsh Transform, Statistical Method

Received: 1 April 2020, Revised 27 July 2020, Accepted 2 August 2020

DOI: 10.6025/jmpt/2020/11/3/95-101

1. Introduction

Bent functions are binary functions with extreme nonlinearity properties. They are actively studied in cryptography, logic synthesis, switching theory, coding theory, and other areas. They become very important for their intensive applications in cryptography.

Bent functions have specific properties and various characterizations. They ensure the cryptographic effectiveness and they can resist to various cryptanalysis attacks. They exist only for the even number of variables. There is no a precise general definition of the structure of bent functions. Also, there is no a formal method for construction of all bent functions. Thus, during recent years, it has been developed a lot of methods for construction subsets of bent functions that have particular properties. The most known methods for construction of bent functions are based on applying combinatorial, algebraic and permutation methods. For example, combinatorial construction methods are Maiorana- McFarland, partial spreads, Dobbertin, iterative constructions, and etc [1]. The most widely known algebraic constructions are monomial bent functions in the Kasami, Gold, Dillon and Canteaut-Leander case, hyper bent functions, Niho bent functions, and etc [2]. The permutations construction methods describes how new bent functions can be obtained from a known bent function, for example in [3].

However, bent functions obtained in this way constitute a small subset of all bent functions, especially for large number of variables [4]. Further, the subsets of bent function generated by proposed deterministic methods do not provide any, for example, cryptographic quality to constructed bent functions.

In cryptographic applications, bent functions need to be nondeterministic. Therefore, bent functions are determined by using a discovery of random bent functions, but the searching time may become prohibitively large when the number of variables is greater than approximately 12. The existing methods are mainly focused on the reduction of the searching time [5].

The most common characterization of bent functions is the equal absolute values of all coefficients of their Walsh spectra. All coefficients has the absolute value $2^{n/2}$ [1].

Testing of all Walsh coefficients requires computation of all 2^n coefficients and related comparisons. This computation, even for small numbers of variables, requires a lot of processing time. Consequently, the number of n- variable Boolean bent functions is known only for $n \leq 8$ [4]. The general number of bent functions is an open problem. Note that, the number of Bent functions increases rapidly with increasing n. Thus, before testing of all Walsh coefficients, the method for bentness testing uses conditional testing of the initial set of particular Walsh coefficients. This initial set of Walsh coefficients usually includes: the first, second, and middle element of the Walsh spectrum.

The efficiency of using conditional testing for particular Walsh coefficients depends on the structure of binary function for which the Walsh spectrum will be calculated. Therefore, in this paper it is proposed an efficient method for bentness testing of binary functions using statistical analysis of binary functions. The statistical analysis of binary functions shows the most efficient selection of the Walsh coefficients for conditional bentness testing. The statistical method is based on the counting of how many times a particular Walsh coefficient of binary n-variable non-bent functions has the absolute value $2^{n/2}$. For the binary functions of many variables, statistical analysis can be applied on the set of 1 million of functions that satisfied bent criteria. The most common bent criteria is restriction of binary function in the spectral Reed-Muller domain, Since the algebraic degree of an n-variable bent function is less or equal to $n/2$, the number of non-zero elements of Reed-Muller spectrum vector is limited and their positions also in the spectrum vector are restricted. Thus, proposed method uses computation of the fast Reed- Muller transform and the computation of the fast Walsh transform.

These statistical results indicate that there is statistically significant relationship between the selection of the Walsh coefficients for conditional bentness testing and efficiency of bentness testing. Experimental results showed that the proposed method can be efficiently used for bentness testing for the functions of 6 to 10 variables.

2. Preliminaries

The Reed-Muller transform [6] represents an important operator for obtaining AND-EXOR expressions of binary functions. The Reed-Muller transform matrix of order n, denoted by $R(n)$, is defined recursively as:

$$R(n) = \bigotimes_{i=1}^{n} R(1),\ R(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tag{1}$$

The Walsh transform [6] is based on a set of orthogonal functions defined by J. L. Walsh which are an extension of a set of functions defined by H. Rademacher. Analogously to previous transforms, the Walsh transform matrix of order n in Hadamard ordering, denoted by $W(n)$, is defined as:

$$W(n) = \bigotimes_{i=1}^{n} W(1),\ W(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2}$$

The spectrum of a binary function f given by truth vector $F = [f(0), f(1),..., f(2^n-1)]^T$ is computed as:

$$S_f = T(n)\ F \tag{3}$$

where $T(n)$ is any of the three matrices $R(n)$, or $W(n)$, with computations performed in $GF(2)$ for the Reed-Muller transform,

form, and in the set of rational numbers for the Walsh transforms.

A binary function $f(x_1, x_2,..., x_n)$ in (1, −1) encoding is called bent if all Walsh coefficients in vector $S_{f,W}$ have the same absolute value $2^{n/2}$ [1].

Algebraic degree of bent functions $f(x_1, x_2,..., x_n)$ in Reed- Muller spectral domain is at most $n/2$ for $n > 2$ [2].

Walsh transform [6] is an important mathematical tool for the analysis of Boolean functions. It can be shown that with (1,−1) encoding of Boolean function values, the Walsh coefficients are even integers in the range 2^{-n} to 2^n. Since the first row of $W(n)$ is equal to constant 1, conditional bentness testing requires calculation of the first Walsh coefficient $S_{f,w}(0)$ expressed as:

$$S_{f,w}(0) = \sum_{i=0}^{2^n-1} F(i) \tag{4}$$

Analogously to previous, second row $W(n)$ takes the successively values +1 and −1. The conditional bentness testing requires calculation of the second Walsh coefficient $S_{f,w}(1)$ expressed as:

$$S_{f,w}(1) = \sum_{i=0}^{2^{n-1}-1} (F(2i) - F(2i+1)) \tag{5}$$

Since the middle row of $W(n)$ takes the value +1 in the first half and −1 in the second half, the conditional bentness testing requires calculation of the middle Walsh coefficient $S_{f,w}(2^{n-1})$ expressed as:

$$S_{f,w}(2^{n-1}) = \sum_{i=0}^{2^{n-1}-1} F(i) - \sum_{i=2^{n-1}}^{2^n-1} F(i) \tag{6}$$

The calculations of first, second and middle Walsh coefficient are exploited in the standard method for the bentness testing of the binary function in the Reed-Muller spectral domain in order to avoid computation of all Walsh coefficients for bent testing.

The recursive definition of the Reed-Muller and the Walsh transform matrices, expressed in Eq. (1), and Eq. (2) respectively, is the fundamental for the definition of fast Reed-Muller and the fast Walsh transform algorithm similar to a fast Fourier transform (FFT) algorithm [6].

The computation of the fast transform algorithm consists of the repeated application of the same "butterfly" operations determined by the basic transform matrices. Figure 1 shows the "butterfly" operations for the Reed-Muller and the Walsh transform matrices [6].

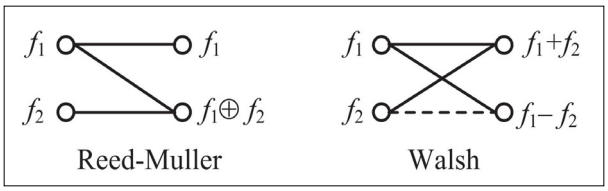

Figure 1. The "butterfly" operations for the Reed-Muller and the Walsh transform matrices

The "butterfly" operations [6] are performed in each step over a different subset of data. Figure 2 shows the flow graphs of the fast Walsh transform algorithm for computation of the Walsh spectrum of a three-variable logic function f given by the truth- vector $F = [f(0), f(1),..., f(7)]^T$.

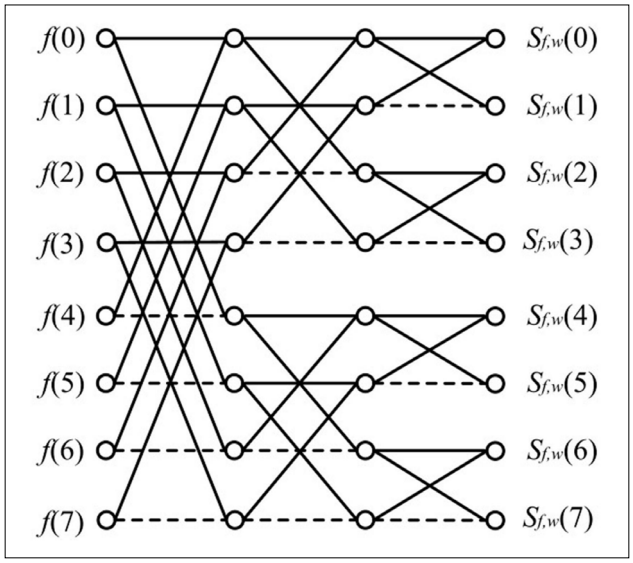

Figure 1. The flow graph of the fast Walsh transform algorithm of the Cooley-Tukey type for the computation of the Walsh spectrum of a three-variable Boolean function

This algorithm is highly exploited for benthess testing in the algortihm for discovering bent functions in the Reed- Muller domain.

3. Statistical Analysis of the Values of Walsh Coefficients

The statistical analysis of the Walsh coefficients values is based on the counting of how many times a particular Walsh coefficient of binary n-variable non-bent functions has the absolute value $2^{n/2}$.

For example, for a binary function of 6 variables, there exist 64 Walsh coefficients. Table 1 shows example of statistical analysis of how many times a particular Walsh coefficient of 6-variable non-bent functions has the absolute value $2^{6/2} = 8$.

The statistical analysis is applied on the testing of set of 1 million binary functions that satisfied bent criteria in Reed-Muller domain. For a binary function of 6 variables, it is evident that statistical analysis shows that first Walsh coefficient for non-bent functions in the least number of cases have absolute value 8. It means that first Walsh coefficient is the most optimal solution for conditional bentness testing. The next Walsh coefficient, that in the least number of cases has absolute value 8, is the last Walsh coefficient. Note that for binary function of 6 variables the difference between statistical values of the first and the last Walsh coefficient is not so high, but it will increases rapidly with increasing number of variables. The next Walsh coefficient, that in the least number of cases has absolute value 8, is the middle Walsh coefficient. In the case of functions of 6 variables it is Walsh coefficient that has index 33.

The statistical analysis of the Walsh coefficients values showed that, initial set of Walsh coefficients need to include: the first, the last, and the middle Walsh coefficient for conditional testing, respectively. It means that before testing of all Walsh coefficients, the method for bentness testing need to perform conditional testing of the first, then testing of the last and finally of the middle Walsh coefficient. The efficiency of the proposed method for bentness testing will be experimentally tested by using a discovery of random bent functions.

4. Method for Bentness Testing of Binary Functions

An outline of the proposed method for the bentness testing of binary functions is given as Algorithm 1.

Note that in previous version of bentness testing methods [7], [8], the conditional testing was done using the first, second, and last coefficient of the Walsh spectrum. The reason for this is that the first, middle and last Walsh coefficient have simple

Index of Walsh coefficient	*Equal to* $\left\|2^{6/2}\right\|$ **for non-bent functions**	*Index of Walsh coefficient*	*Equal to* $\left\|2^{6/2}\right\|$ **for non-bent functions**
1	57108751	33	61302754
2	62280216	34	62684614
3	62280758	35	62685119
4	63193292	36	63201318
5	62156341	37	62658738
6	62818400	38	63216388
7	62818979	39	63216660
8	63104008	40	62467837
9	61287800	41	62604235
10	62656734	42	63299127
11	62657004	43	63299512
12	63192238	44	62534014
13	62736518	45	63376543
14	63201946	46	62613453
15	63201937	47	62613447
16	62440487	48	61723928
17	61302747	48	62616310
18	62684605	50	63316595
19	62685140	51	63316740
20	63201319	52	62545379
21	62658714	53	63397936
22	63216367	54	62624877
23	63216671	55	62625138
24	62467828	56	61741281
25	62604222	57	63475839
26	63299110	58	62690483
27	63299519	59	62690478
28	62534001	60	61805616
29	63376520	61	62714230
30	62613426	62	61882980
31	62613444	63	61882810
32	61723904	64	60913426

Table 1. Example of the statistical analysis of how many times a particular walsh coefficient of 6-variable non-bent functions has the absolute value 8

Algorithm 1 Bentness testing

1: (1, -1) binary function

2: Bent testing of the first Walsh coefficient, if failed go to the step 8.

3: Bent testing of the last Walsh coefficient, if failed go to the step 8.

4: Bent testing of the middle Walsh coefficient, if failed go to the step 8.

5: Transition from truth vector to Walsh spectrum using the fast Walsh transform.

6: Bent testing of all Walsh coefficients, if failed go to the step 8.

7: Return bent function found.

8: Return bent function not found.

formula for calculation and also the implementation is easy.

The proposed method, using results from statistical analysis of the values of Walsh coefficients, made correction in previous approach in step 2. Instead of calculation and conditional testing of the second Walsh coefficient, now the method uses calculation and conditional testing of the last Walsh coefficient.

5. Experimental Results

In this section, the proposed method for bentness testing will be experimentally tested within algorithm for discovering of bent functions with predefined number of non-zero coefficients in Reed-Muller domain. A short outline of the algorithm for discovering bent functions in the RM domain is given as Algorithm 2. The "Bentness testing" algorithm is given as Algorithm 1 within previous section.

Algorithm 2 Discovery of bent function in RM domain

1: Set the number of function variables n and the number of non-zero RM coefficients

2: Random generation of the possible non-zero coefficients in the RM spectrum.

3: Conversion of the random RM spectrum to the binary vector by using the fast RM transform algorithm.

4: If the "Bentness testing" of binary vector is not successful, then go to step 1.

5: Obtain a bent function.

In this paper, the same algorithm for discovering of bent functions in Reed-Muller domain is implemented with standard bentness testing algorithm and also it is implemented with the proposed bentness testing algorithm which is derived form statistical analysis of Walsh coefficients. Experimental results compare the number of discovered functions for the predefined number of non-zero coefficients in RM spectrum. Comparison of these numbers is motivated by increasing the number of discovered bent function when the proposed method is used. Note that discovering of bent function is faster when the number of non-zero coefficients is small.

For experimental purposes, it is developed C++ implementation of Algorithms 1 and 2. The computations are performed on an Intel i7 CPU at 3.66 GHz with 12 GBs of RAM.

Table 2 shows the total number of discovered functions per 1 second for the predefined number of non-zero coefficients in RM domain for standard and proposed method. The data in tables are sorted in the increasing order of the number of nonzero coefficients and the number of variables.

It should be noticed that the number of discovered bent functions of 6, 8, and 10 variables for proposed method is about 3% larger than using the standard method. Note that for the largest binary functions, experiments were not performed, due to very long CPU computation time.

6. Conclusion

This paper proposes an efficient method for bentness testing of binary functions using statistical analysis of the values of Walsh coefficients of binary functions. The standard method for bentness testing before performing the fast Walsh transform calculations uses conditional testing of the initial set of particular Walsh coefficients. The selection of coefficients for conditional testing within standard methods is determined by complexity of calculations.

The conditional testing of the first, the second and the middle Walsh coefficients are exploited in the standard method for bentness testing. The statistical analysis shows that for conditional testing it is more efficient to use the last Walsh coefficient instead of the second. Experimental results confirm that exploiting proposed conditional testing can help to improve the computation performances by 3%.

Num. of function variables	Num of non-zero RM coefficients	Number of discovered bent functions per 1 s	
		standard	proposed
6	10	650	671
6	20	635	653
6	30	627	644
8	10	0.691	0.712
8	20	0.553	0.564
10	10	0.031	0.032
10	20	0.035	0.036

Table 2. Number of discover functions per 1 second for standard and proposed method

We can conclude that, when processing time is a critical parameter, the proposed method for bentness testing should be performed. Future work will be on improving other aspects of the methods for bentness testing.

Acknowledgements

The research reported in this paper is partly supported by the Ministry of Education and Science of Serbia, project III44006 (2011-2019).

References

[1] Tokareva, N. (2015). *Bent Functions, Results and Applications to Cryptography*, Academic Press, 2015.

[2] Mesnager, S. (2016). *Bent Functions, Fundamentals and Results*. Springer International Publishing.

[3] Seberry, J., Zhang, X. (1994). Constructions of bent functions from two known bent functions, *Australasian Journal of Combinatorics*. vol. 9, p. 21-34.

[4] Langevin, P., Leander, G. (2011). Counting all bent functions in dimensions eight 99270589265934370305785861242880, Designs, Codes and Cryptography vol. 59, p. 193-201.

[5] Grocholewska-Czurylo, A., Stoklosa, J. (2002). Generating bent functions, Proc. Advanced Computer Systems Eighth International Conference, ACS2001, p. 361-370. Springer US, Mielno, Poland.

[6] Thornton, M. A., Drechsler, R., Miller, D. M. (2001). *Spectral Techniques in VLSI CAD*, Springer.

[7] Radmanovic, M., Stankovic, R. (2016). Random generation of bent functions on multicore CPU platform. Proc. ICEST 2016, p. 239-242. Ohrid, Macedonia.

[8] Radmanovic, M. (2016). Efficient random generation of bent functions using GPU platform, Proc. 12th Int. Workshop on Boolean Problems, IWSBP2016, p. 167-173. Freiberg, Germany.