Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

On the construction of cryptographically strong Boolean functions with desirable trade-off

REN Kui[1], PARK Jaemin[2], KIM Kwangjo[2]
([1]*ECE Department, Worcester Polytechnic Institute, Worcester, MA 01609, USA*)
([2]*International Research Center for Information Security, Information and Communication University, Daejeon 305-714, Republic of Korea*)
E-mail: kren@ece.wpi.edu; jaemin@icu.ac.kr; kkj@icu.ac.kr

Received May 10, 2004; revision accepted Feb. 11, 2005

Abstract: This paper proposes a practical algorithm for systematically generating strong Boolean functions (f:$GF(2)^n \rightarrow GF(2)$) with cryptographic meaning. This algorithm takes bent function as input and directly outputs the resulted Boolean function in terms of truth table sequence. This algorithm was used to develop two classes of balanced Boolean functions, one of which has very good cryptographic properties: $nl(f)=2^{2k-1}-2^k+2^{k-2}$ ($n=2k$), with the sum-of-squares avalanche characteristic of f satisfying $\sigma_f=2^{4k}+2^{3k+2}+2^{3k}+2^{3k-2}$ and the absolute avalanche characteristic of Δ_f satisfying $\Delta_f=2^{k+1}$. This is the best result up to now compared to existing ones. Instead of bent sequences, starting from random Boolean functions was also tested in the algorithm. Experimental results showed that starting from bent sequences is highly superior to starting from random Boolean functions.

Key words: Boolean functions, Bent sequences, Nonlinearity, GAC, PC, Balancedness
doi:10.1631/jzus.2005.A0358 **Document code:** A **CLC number:** TP301.6

INTRODUCTION

A variety of desirable criteria for functions have been identified: balancedness, local and global avalanche characteristics, high nonlinearity, etc. These properties are also very important for cryptographic purpose. Obtaining optimal tradeoffs among so many properties is hard. If we take into account more criteria, it is more difficult to generate Boolean functions satisfying those properties purely by constructive algebraic methods. How to construct Boolean functions with good properties has received a lot of attention (Clark *et al.*, 2002; Dobbertin, 1995; Millan *et al.*, 1998; 1999; Kim *et al.*, 1991).

A Boolean function f:$GF(2)^n \rightarrow GF(2)$ is called balanced, if the probability that the value of f equals to one is exactly one half, for all possible values of input vector. Webster and Tavares (1985) combined the completeness and avalanche properties into the Strict Avalanche Criterion (SAC). A Boolean function is said to satisfy the SAC if complementing a single bit results in changing the output bit with probability exactly one half. Preneel *et al.*(1990) introduced the propagation criterion of degree k [PC of degree k or $PC(k)$], which generalizes the SAC. A function satisfies $PC(k)$ if by complementing at most k bits the output changes with probability of exactly one half. Although the SAC and PC are very important concepts in designing cryptographic functions employed by encryption and hash functions, they capture only local properties of the function. In order to improve the overall analysis of cryptographically strong functions, Zhang and Zheng (1995) introduced the concepts of Global Avalanche Characteristic (GAC) and proposed two indicators related to GAC: the absolute indicator Δ_f and the sum-of-squares indicator σ_f. Those two criteria overcome the shortcomings of the SAC.

The nonlinearity of a Boolean function f is defined as the minimum Hamming distance from f to all the affine functions defined on $GF(2)^n \rightarrow GF(2)$. It is also an important cryptographic criteria for Boolean functions. Rothaus (1976) showed that for any even n, the maximum nonlinearity achievable for any Boolean function is $2^{n-1}-2^{n/2-1}$. But these functions are not balanced. How to construct balanced Boolean functions on even number of variables with very high nonlinearity was considered by Dobbertin (1995).

As the balanced Boolean functions with good global avalanche characteristics and high nonlinearity can be applied as building blocks of symmetric crypto-systems to resist cryptanalytic attack, it is important to provide an effective and flexible design tool. In the past, most options for Boolean function design have been random generation and direct construction. Both methods have drawbacks. In this paper, we start from bent sequences to construct balanced Boolean functions using an in situ recursive algorithm which can obtain balanced Boolean functions with very good global avalanche characteristics and high nonlinearity.

RELATED WORKS

Zhang and Zheng (1995) proposed two indicators related to GAC: the absolute indicator and the sum-of-squares indicator. The absolute indicator is defined by

$$\Delta_f = \max_{a\in GF(2)^n,\, a\neq 0} \left|\Delta_f(a)\right|$$

and the sum-of-squares indicator by

$$\sigma_f = \sum_{a\in GF(2)^n} \Delta_f^2(2)$$

where $\Delta_f = \sum_{x\in GF(2)^n} (-1)^{f(x)\oplus f(x\oplus a)}$ is the autocorrelation function.

Computing bounds of the two indicators for various classes of Boolean functions is of great importance. The smaller σ_f, Δ_f, the better the GAC of a function. Zhang *et al*. obtained some bounds from the definitions of the two indicators: $2^{2n}\leq\sigma_f\leq 2^{3n}$, $0\leq\Delta_f\leq 2^{3n}$. Son *et al*.(1998) proved that for $n\geq 3$, $\sigma_f\geq 2^{2n}+2^{n+3}$ and $\Delta_f\geq 8$, when f is balanced. Also for balanced Boolean functions, Sung *et al*.(1999) proved that

$$\sigma_f \geq 2^{2n} + 2^6(2^n - t - 1),$$
$$\text{if}\quad 0\leq t\leq 2^n - 2^{n-3} - 1,\quad t\text{=odd}$$
$$\sigma_f \geq 2^{2n} + 2^6(2^n - t + 2),$$
$$\text{if}\quad 0\leq t\leq 2^n - 2^{n-3} - 1,\quad t\text{=even}$$
$$\sigma_f \geq \left(1 + \frac{1}{2^n - t - 1}\right)2^{2n},$$
$$\text{if}\quad 02^n - 2^{n-3} - 1 \leq t \leq 2^n - 2$$

if the function satisfies PC with respect to t vectors.

A function f:$GF(2)^n\rightarrow GF(2)$, n=even number is called bent function, if for any $\omega\in GF(2)^n$, $\hat{F}(\omega)=\pm 2^{n/2}$, where $\hat{F}(\omega)$ is the Walsh transform of function $\hat{f}(\omega)$, i.e., $\hat{F}(\omega)=\sum_{x\in GF(2)^n}\hat{f}(x)(-1)^{\omega x}$ and $\hat{f}(x)=(-1)^{f(x)}$. Note that bent functions have highest nonlinearity when $nl(f)=2^{n-1}-2^{n/2-1}$. The nonlinearity of a Boolean function f is defined as the minimum distance from f to all the affine functions defined on $GF(2)^n\rightarrow GF(2)$, i.e., $nl(f)=2^{n-1}-\max_w|\hat{F}(\omega)|/2$. And also that if a Boolean function is bent, $\sigma_f=2^{2n}$, $\Delta_f=0$ (Zhang and Zheng, 1995).

The problem of constructing Boolean functions which satisfy two or more design criteria seems to be a difficult task. There many works focussed on this area. Among them, many papers started from bent functions to get cryptographically desired Boolean functions, because of their relatively high nonlinearity, good PC and other properties. Meier and Staffelbach (1989) showed how complementing a set of 2^{k-1} ($n=2k$) bits in the truth table of $f(x)$ would yield balanced Boolean functions with good nonlinearity. Stanica (2004) used bent functions to construct "good" Boolean functions with the following properties: $f(x)$ is balanced, and satisfies the SAC when $nl(f)=2^{2k-1}-2^k$, $\sigma_f=2^{4k}+3\times 2^{3k+1}$.

In this paper, we also start from bent sequences to construct balanced Boolean functions using an in situ recursive algorithm with indicator σ_f being used as control parameter. A class of Boolean functions with best trade-offs were obtained with the following properties: $f(x)$ is balanced, and satisfies the propagation criterion with $nl(f)=2^{2k-1}-2^k$, $\sigma_f=2^{4k}+3\times 2^{3k+1}$, $\Delta_f=2^{k+1}$.

STUDY ON THE GAC OF BOOLEAN FUNCTIONS

In this section, we study in which way the GAC property of a given Boolean function can be changed, when one or two bits in its truth table are complemented. We first define $g_k(x)$ and $h_{k_1,k_2}(x)$ to represent the changed Boolean functions when one and two bits of $f(x)$ are changed, respectively.

Definition 1 Let $f(x)$ be a Boolean function defined on $GF(2)^n \rightarrow GF(2)$, with $g_k(x)$ and $h_{k_1,k_2}(x)$ being defined as follows:

$$g_k(x)=\begin{cases} f(x)\oplus 1 & x=x_k \\ f(x) & x\neq x_k \end{cases}$$

$$h_{k_1,k_2}(x)=\begin{cases} f(x)\oplus 1 & x=x_k, k=k_1,k_2 \\ f(x) & x\neq x_k, k=k_1,k_2 \end{cases}$$

where $x\in GF(2)$.

Then by comparing the GAC property of $g_k(x)$ and $h_{k_1,k_2}(x)$ with those of $f(x)$, the change rule of $\Delta_f(a)$ will be obtained.

Theorem 1 Let $g_k(x)$, $h_{k_1,k_2}(x)$ be defined as in Definition 1. Then there will be

$$-\left|\Delta_{g_k}(a)-\Delta_f(a)\right|=4, \text{ for any } a\in GF(2)^n$$

$$-\left|\Delta_{h_{k_1,k_2}}(a)-\Delta_f(a)\right|=8 \text{ or } 0, \text{ for any } a\in GF(2)^n$$

Proof As pointed out by Millan *et al.*(1999), any single truth table change causes $\Delta_f(a)$ changing –2 or 2; any two changes cause changing –4, 4 or 0. So it is easy to find that the results above can be proved.

The following theorem shows how to modify the value of $\Delta_f(a)$ of a Boolean function that has been altered in a single truth table position, with complexity $O(2^n)$.

Theorem 2 Let $g_k(x)$ be defined as in Definition 1. Then each value of $\Delta_g(a)$, $\Delta_g(a)=\Delta_f(a)+\Delta(a)$, can be obtained as follows: if $f(x_k)=f(x_k\oplus a)$, then $\Delta(a)=-4$, else $\Delta(a)=+4$.

Proof When $f(x_k)=f(x_k\oplus a)$, it follows that $(-1)^{f(x_k)+f(x_k\oplus a)}=1$, the square of which contributes to the sum of σ_f. Changing the value of $f(x_k)$ changes this contribution to −1 in two places, so $\Delta_g(a)-\Delta_f(a)=-4$. Similarly, when $f(x_k)\neq f(x_k\oplus a)$, $\Delta_g(a)-\Delta_f(a)=+4$.

Now considering the definition of the sum-of-squares indicator, we can see that the sum-of-square indicator $\sigma_g/(\sigma_h)$ of $g_k(x)/(h_{k_1,k_2}(x))$ can be derived from σ_f. We first divide the value space of a into the following subsets as Definition 2 shows.

Definition 2 A Boolean function $f(x)$ is defined on $GF(2)^n \rightarrow GF(2)$. Define Δ^+, Δ^- and Δ^0 as follows:

$$\Delta^+=\{a: \Delta_f(a)>0\},\ \Delta^-=\{a: \Delta_f(a)=0\},\ \Delta^0=\{a: \Delta_f(a)=0\}.$$

In order to compute the number of times when equation $f(x)=f(x\oplus a)$ holds within Δ^+ and Δ^-, define Δ_1^+ and Δ_1^- as follows:

Definition 3 Let x_k be a constant input vector defined on $GF(2)^n$. Define Δ_1^+ and Δ_1^- as follows:

$$\Delta_1^+=\left\{a: f(x_k)=f(x_k\oplus a), a\in\Delta^+\right\}$$

$$\Delta_1^-=\left\{a: f(x_k)\neq f(x_k\oplus a), a\in\Delta^-\right\}$$

Definition 4 Let $g_k(x)$ be defined as in Definition 1. If $\#\{\Delta_1^+\cup\Delta_1^-\}>2^{n-1}$, then we say that is a 'good change' of $f(x)$. Define the 'goodness' of $g_k(x)$ as $G(g)$:

$$G(g)=\#\{\Delta_1^+\cup\Delta_1^-\}-2^{n-1}$$

in order to evaluate how 'good' $g_k(x)$ is. A function with larger $G(g)$ is said to be 'better' than another one.

Definition 5 Let x_{k-1} and x_{k-2} be two constant input vectors both defined on $GF(2)^n$. Define Δ_0^+, Δ_0^-, Δ_2^+, Δ_2^-, Δ_3^+ and Δ_3^- as follows:

$$\Delta_0^+=\left\{a: f(x_k)=f(x_k\oplus a), k=k_1,k_2, a\in\Delta^0\right\}$$

$$\Delta_0^-=\left\{a: f(x_k)\neq f(x_k\oplus a), k=k_1,k_2, a\in\Delta^0\right\}$$

$$\Delta_2^+=\left\{a: f(x_k)=f(x_k\oplus a), k=k_1,k_2, a\in\Delta^+\right\}$$

$$\Delta_2^-=\left\{a: f(x_k)\neq f(x_k\oplus a), k=k_1,k_2, a\in\Delta^-\right\}$$

$$\Delta_3^+=\left\{a: f(x_k)=f(x_k\oplus a), k=k_1,k_2, a\in\Delta^+\right\}$$

$$\Delta_3^-=\left\{a: f(x_k)\neq f(x_k\oplus a), k=k_1,k_2, a\in\Delta^-\right\}$$

Definition 6 Let $h_{k_1,k_2}(x)$ be defined as in Definition 1. If

$$\#\{\Delta_2^+ \cup \Delta_2^-\} > \#\{\Delta_3^+ \cup \Delta_3^- \cup \Delta_0^+ \cup \Delta_0^-\}$$

holds, we say that $h_{k_1,k_2}(x)$ is a 'good change' of $f(x)$.

Define the 'goodness' of $h_{k_1,k_2}(x)$ as $G(h)$:

$$G(h) = \#\{\Delta_2^+ \cup \Delta_2^-\} - \#\{\Delta_3^+ \cup \Delta_3^- \cup \Delta_0^+ \cup \Delta_0^-\}$$

in order to evaluate how 'good' $h_{k_1,k_2}(x)$ is. A function with larger $G(h)$ is said to be 'better' than another one.

Obviously, the definition of 'goodness' of $h_{k_1,k_2}(x)$ can lead to an explicit evaluation of $h_{k_1,k_2}(x)$. Thus different $h_{k_1,k_2}(x)$ can be compared with each other using "goodness" as the parameter.

THE IN SITU RECURSIVE ALGORITHM

In this section, we describe our in situ recursive algorithm in detail. A technique called hill climbing was introduced by Millan *et al.*(1997). The basic idea of hill climbing is to slightly alter a given Boolean function so that the property of interest, such as nonlinearity, can be improved. Their results showed that hill climbing could considerably improve the nonlinearity of randomly generated Boolean functions. Based on this basic idea, we come up with our in situ recursive algorithm to achieve better results. Many properties of interests, such as nonlinearity, GAC property, are considered at the same time. In our algorithm, the change to the truth table is determined by the parameter "goodness" defined in the above section. This is critical to the final results we can obtain. Instead of starting from random Boolean functions, bent function is chosen as the start point.

As mentioned in Section 1, bent functions have very good properties, such as nonlinearity and GAC property. But bent functions are not balanced. Rothaus (1976) proved that every bent sequence of length 2^{2k} had hamming weight $2^{2k-1}\pm 2^{k-1}$ and proved the following theorem.

Theorem 3 For an even number n of arguments bent functions are constructed as follows: Let $n=2m$. Then functions of the form $f(x_1, x_2, \ldots, x_n)=g(x_1, x_2, \ldots, x_m)+x_1x_{m+1}+x_2x_{m+2}+\ldots+x_mx_n$ are bent, where $g(x_1, x_2, \ldots, x_m)$ is a completely arbitrary function of m variables.

Obviously, Theorem 3 leads to an explicit construction of bent functions, though many other construction methods also exist. In our algorithm, we use this method to construct initially bent functions.

Simply changing 2^{k-1} bits within the bent sequence can obviously cause the function to be balanced. But it is computationally intensive to exhaustively alter all $C_{2^{2k-1}+2^{k-1}}^{2^{k-2}}$ truth table positions and compute the value of control indicator σ_f every time. A fast systematic method is adopted in our algorithm instead of this cumbersome one. The method complements 2-bit each time until the sequence is balanced. It is easy to see that there are only $C_{2^{2k-1}+2^{k-1}}^2 \cdot C_{2^{2k-1}+2^{k-1}-2}^2 \ldots C_{2^{2k-1}+2^{k-2}}^2$ rounds needed. And the computation of σ_f needs to be executed only once.

A new approach is used to determining bit positions in a function's truth table during each round. The two positions in the truth table to be changed are chosen according to the value of 'goodness' as defined in Definition 6. It is important to note that in this way we can avoid computing the value of σ_f, which is highly time consuming.

After the sequence is balanced, we can then compute the value of σ_f. We will then check whether the GAC property can be improved while maintaining the balancedness of the resulted sequence. This means we will choose two bits with opposite value and then complement these two bits. The results indicated that no improvement can be made to the resulted sequence.

So the in situ recursive algorithm takes as its input a Boolean functions binary truth table. A bent sequence is firstly generated as the input of the in situ recursive algorithm using Theorem 3. The algorithm then tests every two-bit pairs each time within the bent sequence to find a $h_{k_1,k_2}(x)$ with highest 'goodness', and then complements the bent sequence at the positions given by resulted k_1 and k_2. Use the newly obtained sequence as the current input of the algorithm and repeat the same procedure, until the sequence is balanced.

The following is the pseudo code of the in situ recursive algorithm.

```
In situ recursive algorithm:
1. Generate a $2k$-variable bent sequence ($k\geq 3$) using the above mentioned Theorem 3. Compute $\Delta_f$, $a\in GF(2)^n$, $a\neq 0$, and $\sigma_f$, and store them. Compute hamming weight $\omega t(f)$. According to the value of $\Delta_f$, create $\Delta^+$, $\Delta^-$ and $\Delta^0$.
2. If $\omega t(f)=2^{2k-1}$, then go to Step 8.
3. If $\omega t(f)>2^{2k-1}$, then go to Step 6.
4. For $k_1=1, \ldots, 2^n-1$ and $f(k_1)=0$, do
   a) for $k_2=k_1+1, \ldots, 2^n-1$ and $f(k_2)=0$, do
      i. Compute the value of
         $X: \#\{\Delta_3^+ \cup \Delta_3^- \cup \Delta_0^+ \cup \Delta_0^-\} - \#\{\Delta_2^+ \cup \Delta_2^-\}$.
      ii. Compare $X$ with currently smallest one and keep the smaller one as well as its bit positions.
      iii. $k_2=k_2+1$
   b) $k_1=k_1+1$
5. Complement the resulted bit positions, update $\omega t(f)=\omega t(f)-4$, and update the value of $\Delta_f$, by applying Theorem 2 twice. Update $\Delta^+$, $\Delta^-$ and $\Delta^0$. And go to Step 2.
6. For $k_1=1, \ldots, 2^n-1$ and $f(k_1)=1$, do
   a) for $k_2=k_1+1, \ldots, 2^n-1$ and $f(k_2)=1$, do
      i. compute the value
         $X: \#\{\Delta_3^+ \cup \Delta_3^- \cup \Delta_0^+ \cup \Delta_0^-\} - \#\{\Delta_2^+ \cup \Delta_2^-\}$.
      ii. Compare $X$ with currently smallest one and keep the smaller one as well as its bit positions.
      iii. $k_2=k_2+1$
   b) $k_1=k_1+1$
7. Go to Step 5.
8. Compute and output the value of $\sigma_f$. Output the resulted sequence.
```

EXPERIMENT RESULTS AND DISCUSSION

Two classes of solutions are systematically generated by the proposed algorithm, namely Class 1 and Class 2, respectively. It was found that Class 2 holds the same properties as the results obtained by Zhang and Zheng (1995) by construction on V_{2k} using Eq.(5). In other word, these Boolean functions are of the same kind. And Class 1 obviously outperforms Class 2 with higher nonlinearity and smaller the sum-of-squares indicator σ_f. We have following results.

Result Let the results be given by the above in situ algorithm. Then will be either Class 1 or Class 2:

Class 1:

1. f is balanced,
2. The nonlinearity of f is $nl(f)=2^{2k-1}-2^k+2^{k-2}$,
3. The sum-of-squares indicator of f is $\sigma_f=2^{4k}+2^{3k+2}+2^{3k}+2^{3k-2}$,
4. The absolute indicator of f is $\Delta_f=2^{k+1}$,
5. f satisfies the propagation criterion with respect to $2^{2k}-2^{k+1}+2^{k-3}-1$ nonzero vectors.

Class 2:

1. f is balanced,
2. The nonlinearity of f is $nl(f)=2^{2k-1}-2^k$,
3. The sum-of-squares indicator of f is $\sigma_f=2^{4k}+2^{3k+2}+2^{3k+1}$,
4. The absolute indicator of f is $\Delta_f=2^{k+1}$,
5. f satisfies the propagation criterion with respect to $2^{2k}-2^{k+1}+2^{k-1}-1$ nonzero vectors.

We now present examples of the results obtained by this algorithm with n=6, 8, 10, 12, respectively. Table 1 shows the sample results (Class 1). Note that the test results indicated that all the Boolean functions

Table 1 Sample results of the in situ recursive algorithm

n	6		8		10		12	
	1	2	1	2	1	2	1	2
Balancedness	yes	yes	yes	yes	yes	yes	yes	yes
Nonlinearity	26	24	116	112	488	480	2000	1984
σ_f	6784	7168	87040	90112	1220608	1245184	18153472	18350080
Δ_f	16	16	32	32	64	64	128	128
Nonzero vectors not satisfying PC	15	12	30	24	60	48	120	96

with the same n are different from each other.

Note that Class 2 appears more frequently than Class 1. Fig.1 shows the proportion of two kinds of solutions when n=6, 8, respectively. And it seems that the appearance of Class 1 depends on its start bent function, though the relationship between them is not explicit.

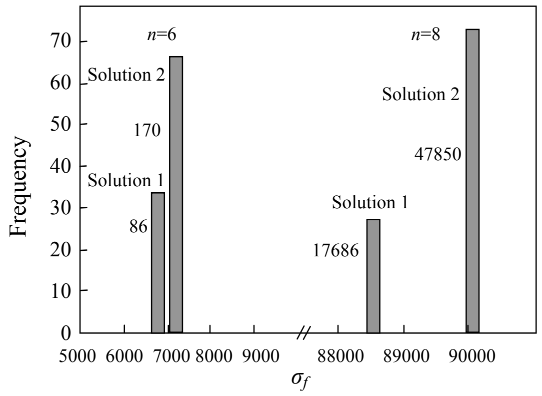

Fig.1 The proportion of Class 1 vs Class 2

Clark *et al.*(2002) compared the nonlinearity of balanced Boolean functions as Table 2 shows. Compared with other results provided in Table 2, the nonlinearity character of Class 1 is very good, and can be systematically generated.

Table 2 Comparing the nonlinearity of balanced Boolean function

n	6	8	10	12
Lowest upper bound	26	118	494	2014
Best known example (Hou, 1993; Patterson and Wiedemann, 1983)	26	116	492	2010
Dobbertin's conjecture (Dobbertin, 1995)	26	116	492	2010
Bent concatenation	24	112	480	1984
Random generated	–	112	472	1954
Hill climbing algorithm	–	114	476	1960
Genetic algorithm (Millan *et al.*, 1998)	26	116	484	1976
NLT (Clark *et al.*, 2002)	26	116	486	1992
ACT (Clark *et al.*, 2002)	26	116	484	1986
Our algorithm	26	116	488	2000

Many works aimed at constructing Boolean functions with very good GAC properties. Some of them are listed in Table 3 for the comparison with Class 1.

Clark *et al.*(2002) used a simulated annealing method to search and find cryptographically good Boolean functions described in Section 3, Table 4 shows that when n=8, this paper got better result on the value of σ_f and that our result outperforms their results when n=10, and that they did not provide any results when n>10.

Table 3 GAC properties comparison

Method	GAC properties	
	σ_f	Δ_f
Our algorithm	$2^{4k}+2^{3k+2}+2^{3k}+2^{3k-2}$	2^{k+1}
Zhang and Zheng (1995)	$2^{4k}+3\cdot2^{3k+1}$	2^{k+1}
Stanica and Sung (2001)	$2^{4k+1}\le\sigma_f\le2^{4k+2}$	2^{k+1}
Stanica (2004)	$2^{4k}+3\cdot2^{3k+1}$	2^{k+1}
Stanica (2002)	2^{4k+2}	2^{k+1}

Table 4 Properties of randomly generated Boolean functions

n	6	8	10	12
Balancedness	yes	yes	yes	yes
Nonlinearity	≤22	≤106	≤464	≤1940
σ_f	$>10^4$	$>17\times10^4$	$>34\times10^5$	$>46\times10^6$
Δ_f	≥24	≥48	≥136	≥288
Nonzero vectors not satisfying PC	>40	>206	>929	>3900

For comparison, we also started from random generated balanced Boolean functions to search for strong Boolean functions. Table 4 shows the properties of randomly generated Boolean functions. Similarly, truth table modifying methods were used while keeping balancedness. We found that all properties mentioned above were far from good. For example, when n=8, the value of σ_f is typically larger than 100000, and $\Delta_f\ge40$, with around 160 nonzero vectors not satisfying PC. And the nonlinearity was typically around 110. No Boolean functions were found as good as the better results obtained by our algorithm.

CONCLUSION

We have presented a highly efficient in situ recursive algorithm to find strong Boolean functions for cryptographic applications, and this algorithm can be a good designing tool of cryptographically good Boolean functions. With this algorithm we have obtained currently best trade-off with $nl(f)=2^{2k-1}-2^k+2^{k-2}$ and the sum-of-squares avalanche characteristic of f satisfies $\sigma_f=2^{4k}+2^{3k+2}+2^{3k}+2^{3k-2}$, the absolute avalanche characteristic of f satisfies $\Delta_f=2^{k+1}$.

Several open questions remain. One is why only two kinds of results exist. The relationship between our results and bent functions constructed by Theorem 3 should be further explored. Another is whether Class 1 can lead to a construction of S-Box with nonlinearity larger than 80, which is currently the highest. Researches on these topics are ongoing.

References

Clark, J.A., Jacob, J.L., Stepney, S., Maitra, S., Millan, W., 2002. Evolving Boolean Functions Satisfying Multiple Criteria. International Conference on Cryptology in India–INDOCRYPT 2002, LNCS 2551, Springer-Verlag, p.246-259.

Dobbertin, H., 1995. Construction of Bent Functions and Balanced Boolean Functions with High Nonlinearity. Fast Software Encryption–FSE'94, LNCS 1008, Springer-Verlag, p.61-74.

Hou, X.D., 1993. Further results on the covering radii of the Reed-Muller codes. *Design, Codes and Cryptography*, **3**(2):167-177.

Kim, K., 1991. Construction of DES-like S-boxes Based on Boolean Functions Satisfying the SAC. Advances in Cryptology–Proc. of Asiacrypt'91, Lecture Notes in Computer Science, Springer-Verlag, p.59-72.

Meier, W., Staffelbach, O., 1989. Nonlinearity Criteria for Cryptographic Functions. Advances in Cryptology–EUROCRYPT'89, LNCS 434, Springer-Verlag, p.549-562.

Millan, W., Clark, A.J., Dawson, E., 1997. Smart Hill Climbing Finds Better Boolean Functions. Workshop on Selected Areas in Cryptology 1997, Workshop Record, p.50-63.

Millan, W., Clark, A.J., Dawson, E., 1998. Heuristic Design of Cryptographically Strong Balanced Boolean Functions. Advance in Cryptology–EUROCRYPT'98, LNCS 1403, Springer-Verlag, p.489-499.

Millan, W., Clark, A., Dawson, E., 1999. Boolean Function Design Using Hill Climbing Methods. Australasian Conference on Information Security and Privacy–ACISP'99, LNCS 1587, Springer-Verlag, p.1-11.

Patterson, N.J., Wiedemann, D.H., 1983. The covering radius of the (2^{15},6) Reed-Muller code is at least 16276. *IEEE Transactions on Information Theory*, **IT-29**(3):354-356.

Preneel, B., Van Leekwijck, W., Van Linden, L., Govaerts, R., Vandewalle, J., 1990. Propagation Characteristics of Boolean Functions. Advances in Cryptology–EUROCRYPT'90, LNCS 473, Springer-Verlag, p.161-173.

Rothaus, S., 1976. On bent functions. *Journal of Combinatorial Theory, Ser. A*, **20**:300-305.

Son, J.J., Lim, J.I., Chee, S., Sung, S.H., 1998. Global avalanche characteristics and nonlinearity of balanced Boolean functions. *Information Processing Letters*, **65**(3): 139-144.

Stanica, P., 2002. Nonlinearity, local and global avalanche characteristics of balanced Boolean functions. *Discrete Mathematics*, **248**(1-3):181-193.

Stanica, P., 2004. Boolean functions with five controllable cryptographic properties. *Designs, Codes and Cryptography*, **31**:147-157.

Stanica, P., Sung, S.H., 2001. Improving the nonlinearity of certain balanced Boolean functions with good local and global avalanche characteristics. *Information Processing Letters*, **79**(4):167-172.

Sung, S.H., Chee, S., Park, C., 1999. Global avalanche characteristics and propagation criterion of balanced Boolean functions. *Information Processing Letters*, **69**(1):21-24.

Webster, F., Tavares, S.E., 1985. On the Design of S-boxes. Advances in Cryptology–CRYPTO'85, LNCS 218, Springer-Verlag, p.523-534.

Zhang, X., Zheng, Y.L., 1995. GAC–the criterion of global avalanche characteristics of cryptographic functions. *Journal of Universal Computer Science*, **1**(5):316-333.