Heuristic Design of Cryptographically Strong Balanced Boolean Functions

William Millan, Andrew Clark and Ed Dawson

Information Security Research Center,
Queensland University of Technology,
GPO Box 2434, Brisbane, Queensland, Australia 4001.
FAX: +61-7-3221 2384
Email: {millan,aclark,dawson}@fit.qut.edu.au

Abstract. Advances in the design of Boolean functions using heuristic techniques are reported. A genetic algorithm capable of generating highly nonlinear balanced Boolean functions is presented. Hill climbing techniques are adapted to locate balanced, highly nonlinear Boolean functions that also almost satisfy correlation immunity. The definitions for some cryptographic properties are generalised, providing a measure suitable for use as a fitness function in a genetic algorithm seeking balanced Boolean functions that satisfy both correlation immunity and the strict avalanche criterion. Results are presented demonstrating the effectiveness of the methods.

1 Introduction

It is well known that the resistance of a product cipher to modern cryptanalytic attacks such as linear and differential cryptanalysis [10,1] depends critically upon the nonlinearity of the Boolean functions comprising the round function. Typically these functions must be balanced, so there is considerable interest in the design of highly nonlinear balanced Boolean functions. In addition we would like cipher functions to satisfy other cryptographic properties also, such as correlation immunity [20] and the strict avalanche criterion (SAC) [25]. Previous work on the design of balanced functions includes [6,7,17–19,21]. The existing body of research concentrates on specific constructions, supported by algebraic proofs that the resulting Boolean functions will be both balanced and satisfy one or more other properties. In contrast, some recent publications [13,12] address the issue of applying combinatorial optimisation methods to the design of Boolean functions. The methods of gradient descent (or *hill climbing* (HC)) and the use of a genetic algorithm have proven useful in the quasi-random generation of highly nonlinear Boolean functions.

In this paper we present a modification of the genetic algorithm (GA) presented in [12] so that it is confined to balanced Boolean functions. When combined with the two-step hill climbing algorithm [13] a very effective means of generating highly nonlinear balanced Boolean functions is obtained. The results

presented in Section 4 show clearly that our hybrid algorithm is far more effective than blind search in finding balanced Boolean functions with a high nonlinearity.

We observe that the hill climbing technique can be adapted to find t-resilient functions, which are both balanced and correlation immune of order t ($CI(t)$). We describe a new algorithm and give performance results. In particular, the nonlinearity of functions obtained by the method is of primary interest.

Finally a genetic algorithm is presented which seeks functions that are balanced, and satisfy both $CI(1)$ and the SAC, which is equivalent to the propagation criterion of order 1, $PC(1)$ [16]. The algorithm is described in Section 3 and its performance is discussed in Section 4. Now we present a review of some cryptographic properties of Boolean functions, and propose a definition for the deviation a function has from the strict properties of correlation immunity and the propagation criteria.

2 Some Properties of Boolean Functions

Consider an n variable Boolean function $f(x) : Z_2^n \to Z_2$. The truth table of the function is a list of 2^n bits representing the output of the function for each input. Let the linear function selected by ω be $L_\omega(x) = \omega_1 x_1 \oplus \omega_2 x_2 \oplus \cdots \oplus \omega_n x_n$. It is useful to represent a Boolean function in polarity form, replacing the symbols $\{0, 1\}$ with $\{1, -1\}$. We define the polarity truth table as $\hat{f}(x) = (-1)^{f(x)}$. Similarly we have $\hat{L}_\omega(x) = (-1)^{L_\omega(x)}$. With this representation the Walsh-Hadamard transform of the function can be defined as $\hat{F}(\omega) = \sum_x \hat{f}(x)\hat{L}_\omega(x)$. The maximum absolute value taken by the Walsh-Hadamard transform over all ω has been called the *spectral radius* in [6]. Here we will denote it as WH_{max}. It is well known that the nonlinearity of $f(x)$ is given by $N_f = \frac{1}{2} * (2^n - WH_{max})$. To increase the nonlinearity of a Boolean function, WH_{max} must decrease.

The autocorrelation function is defined as $\hat{r}(s) = \sum_x \hat{f}(x)\hat{f}(x \oplus s)$. It is well known [15] that the autocorrelation function can be efficiently calculated from the square of the Walsh-Hadamard representation by performing an inverse Walsh-Hadamard transform.

Theorem 1 (Wiener-Kintchine). *Let a Boolean function $f(x)$ have Walsh-Hadamard transform $\hat{F}(\omega)$ and autocorrelation function $\hat{r}(s)$. For all $\omega \in Z_2^n$ it is true that*

$$\sum_{s \in Z_2^n} \hat{r}(s)(-1)^{s \cdot \omega} = \left(\hat{F}(\omega)\right)^2.$$

Proof: Omitted. For example see [2].

The properties of correlation immunity and the propagation characteristics can be most easily defined in terms of $\hat{F}(\omega)$ and $\hat{r}(s)$. These properties have received considerable attention in the literature, and are well known. A Boolean function $f(x)$ is said to satisfy correlation immunity of order m if and only if $\hat{F}(\omega) = 0$ for all those ω with $1 \leq |\omega| \leq m$ [26]. Similarly a Boolean function $f(x)$ is said to satisfy the propagation criterion order k if and only if $\hat{r}(s) = 0$,

for all those s with $1 \leq |s| \leq k$ [16]. We note that balanced $CI(t)$ functions are also called t-resilient [24]. Balanced $PC(k)$ functions have no special name.

We now present the definitions of *deviation* from these properties. The reason for this approach is twofold: firstly to provide a quantitative means of assessing functions that almost satisfy the conditions for $CI(m)$ and $PC(k)$ and secondly, to provide a suitable measure of fitness for a genetic algorithm seeking these functions.

Definition 2. The correlation immunity deviation of a Boolean function $f(x)$ is defined as

$$cidev_f(m) = \max(|\hat{F}(\omega)|; 1 \leq |\omega| \leq m).$$

We note that $cidev(n) = WH_{max}$ for all Boolean functions. Any function with $cidev(m) = 0$ satisfies $CI(m)$.

Definition 3. The propagation characteristic deviation of a Boolean function $f(x)$ is defined as

$$pcdev_f(k) = \max(|\hat{r}(s)|; 1 \leq |s| \leq k).$$

Any function with $pcdev(k) = 0$ satisfies $PC(k)$.

The deviation measures provide a meaningful way to describe functions that do not quite satisfy the strict properties. We can also consider the extent to which functions approach CI and PC simultaneously.

Definition 4. The normalised deviation of a Boolean function $f(x)$ is defined as

$$normdev_f(m,k) = \max\left\{\frac{cidev_f(m)}{2}, \frac{pcdev_f(k)}{4}\right\}$$

The normalised deviation has been used as the fitness function in genetic algorithms seeking Boolean functions that are balanced, $CI(1)$ and $PC(1)$. The performance results of this approach are presented in Section 4. We now briefly discuss the known bounds on the nonlinearity of balanced functions before turning to descriptions of the GA and HC algorithms.

2.1 Nonlinearity of Balanced Functions

It is known that the maximum nonlinearity of Boolean functions corresponds to the covering radius problem for Reed-Muller Codes [9]. Thus it is known that for n even, the maximum nonlinearity attainable is $N_{max}(n) = 2^{n-1} - 2^{\frac{n}{2}-1}$, but such functions (bent functions) are not balanced. For odd n, the situation is less certain. It is known that for $n = 3, 5$ and 7 the maximum nonlinearity is $2, 12$ and 56, respectively [14]. For odd $n \geq 9$ no tight bounds are known. An upper bound, which is known to be wide [18], states that the nonlinearity of a balanced Boolean function satisfies $N \leq 2^{n-1} - 2^{\frac{n}{2}-1} - 2$ for n even. A recent paper [8] gave a slightly improved upper bound for the nonlinearity when n is

odd: $N \leq 2\lfloor 2^{n-2} - 2^{\frac{n}{2}-2} \rfloor$. However this bound is known to be not tight for $n = 7$ and $n = 15$. For odd $n \geq 9$ it is an open problem to find the true upper bound.

Some Boolean function constructions are known to generate examples approaching these bounds. For example, it is well known that, by concatenating bent functions, for n odd it is possible to construct a balanced function satisfying $N = 2^{n-1} - 2^{\frac{n-1}{2}}$, and for even n a balanced function with nonlinearity $N = 2^{n-1} - 2^{\frac{n}{2}}$ can be constructed. However, in general, the maximum nonlinearity attainable by balanced functions is not known. The work presented in [18] and [6] has shown that the attainable bound for nonlinearity of balanced Boolean functions can exceed the value attained by concatenation of bent functions. It was shown in [18] that by slightly modifying the functions being concatenated that higher values of nonlinearity for balanced functions could be achieved. The values are tabulated for various n. We note that achieved values correspond exactly to the upper bounds implied by Conjecture A in [6], for the listed n. A summary of these results is shown in Table 1. The highest nonlinearity found by our hybrid genetic/hill climbing algorithm (in experiments so far) is presented for comparison with the lowest known upper bounds given by theory, and the best known examples. Where the lowest theoretical bound exceeds the best example known, it remains an open problem to determine the value of the true upper bound.

Method	4	5	6	7	8	9	10	11	12
Lowest Upper Bound [14,8]	4	12	26	56	118	244	494	1000	2014
Best Known Example [18]	4	12	26	56	116	240	492	992	2010
Bent Concatenation	4	12	24	56	112	240	480	992	1984
Our Genetic Algorithm	4	12	26	56	116	236	484	980	1976

Table 1. Comparing the Nonlinearity of Balanced Functions

The main open problems surrounding the nonlinearity of balanced Boolean functions are
(1) for $n = 8$, is 116 or 118 the maximum balanced nonlinearity?
(2) for $n = 9$ can 240 be exceeded? In principle our heuristic algorithms, given large enough pool sizes and sufficiently many iterations, could provide examples of balanced Boolean functions at the true upper bound. Several of the known bounds have been achieved during our experimenmts. However, no new bounds have yet been obtained by these methods.

3 The Genetic Algorithm

Genetic algorithms (GAs) have been successfully applied to numerous applications in the field of optimisation. They have also been used as a tool for cryptanalysis – with varying degrees of success. The classical ciphers are typically

vulnerable to attacks from GAs, for example [23,11,5,3]. However, an attack presented in [22] on knapsack-type ciphers was found to be flawed [4].

The genetic algorithm borrows concepts from the evolutionary process (such as "survival of the fittest" and "genetic mutation") to *breed* a *pool* of solutions which are considered most *fit*. Traditionally binary solution structures are used however, *evolutionary programming* encompasses arbitrary solution structures. The truth table representation of a Boolean function provides a suitable binary solution structure which has been utilised throughout all the experiments reported here.

A typical GA combines processes of selection, breeding and mutation. There must exist a mechanism for evaluating arbitrary solutions – called the *fitness function*. In this paper the three main evaluation criteria used are the nonlinearity, which is maximized, the deviation from correlation immunity ($cidev(1)$) and the normalised deviation from $CI(1)/PC(1)$ ($normdev(1,1)$) both of which are minimized. In addition to these criteria we impose the restriction that the Boolean function must be balanced since this is desirable in most cryptographic applications. In GAs seeking to find a good compromise between high nonlinearity and low $CI(m)$ deviation, we have found that maximizing $nonlin - cidev(m)$ is more effective than using either criteria alone.

Initially, a pool of P solutions is chosen and the fitness of each solution in the pool is calculated. Here, the pool consists of truth tables corresponding to (initially random) balanced Boolean functions. From this pool pairs of parents are chosen to act as the parents of the next generation. Parents may be chosen randomly, based on their fitness or (as in this case) exhaustively (all possible pairings are tried). The breeding process requires some mating function for combining parent solutions. Here we use a merging operation which combines two parents to produce a single offspring. The offspring will be a balanced function which is similar to each of its parents (the merge operation is described in detail below). Typically, each of the offspring undergo some mutation. As will be seen below, the merging operation used incorporates a random mutation so a separate mutation operation is not required. At this stage the survivors for the next iteration are chosen. This involves combining the parent and offspring pools and selecting the most fit as the new solution pool for the next iteration.

The merging (or mating) operation is now described. This operation takes two balanced Boolean functions as input and produces a single balanced Boolean function as the offspring. Consider two Boolean functions of n inputs. The truth tables corresponding to these functions will contain 2^n bits. Call the two parent functions p_1 and p_2, and let $p_k[i]$ denote the i^{th} bit in the truth table of parent k. Also, n_1 denotes the number of 1's which have been placed in the child in positions where the parents differ, and $\text{dist}(p_1, p_2)$ is the Hamming distance between the two truth tables, p_1 and p_2. The objective of the algorithm is to ensure that a child is produced that satisfies $n_1 = \frac{1}{2}\text{dist}(p_1, p_2)$, since this ensures that the child is balanced. The offspring c is determined as follows:

1. Let $n_1 = 0$ and $k = 0$.
2. If $\text{dist}(p_1, p_2) > 2^n/2$ complement p_1 or p_2.

3. For $i = 1$ to 2^n do:
   (a) If $p_1[i] = p_2[i]$ then $c[i] = p_1[i]$ $(= p_2[i])$;
   (b) Otherwise (if $p_1[i] \neq p_2[i]$)
      i. If $n_1 = \text{dist}(p_1, p_2)/2$ then $c[i] = 0$;
      ii. Else if $n_1 + \text{dist}(p_1, p_2) - k = \text{dist}(p_1, p_2)/2$ then $c[i] = 1$;
      iii. Else $c[i] =$ a random bit.
      iv. Increment k $(k = k + 1)$.
      v. If $c[i] = 1$ then $n_1 = n_1 + 1$.

The check in Step 2 is to ensure that only parents which are close to each other are allowed to breed. It should be noted that complementing a Boolean function's truth table does not alter its nonlinearity. The checks in Steps 3(b)i and 3(b)ii are used to force the offspring to be balanced.

The overall genetic algorithm could then be described as follows:

1. Generate a pool of P random, balanced Boolean functions (represented by their truth table) and calculate the fitness of each. Call this pool S_0.
2. For $i = 1$ to MAXITER do:
   (a) For all $P(P-1)/2$ pairings of the pool S_{i-1} perform the merging operation (as described above) to produce $P(P-1)/2$ offspring.
   (b) (Optional hill climbing.) Apply the hill climbing procedure to each of the offspring.
   (c) Determine the fitness of each of the offspring.
   (d) Combine the pool of offspring with the current pool, S_{i-1}, and select the P best as the new pool, S_i. Give precedence to offspring with fitness equal to solutions already in the pool. Duplicate solutions should be removed.
   (e) (Optional *resetting* step.) If there has been no improvement in the fitness of the best solution for a number of iterations, then keep the best solution and generate $P - 1$ random, balanced functions as the remainder of the pool.
3. Output the best solution from the current pool.

The hill climbing procedure referred to above was described in [13]. The optional resetting step was found to enhance the algorithm when hill climbing is not used. The resetting step essentially randomises the pool after it has converged, but retains the best solution. Figure 1 gives a comparison of the performance of GAs with and without resetting and hill climbing. It can be seen that for algorithms without hill climbing the resetting technique provides improvement. It is noted that the resetting process is not effective (no improvement was gained in our experiments) for genetic algorithms which incorporate hill climbing.

The hill climbing procedure in [13] provides a method of determining which bits in a function's truth table can be complemented in order to improve the nonlinearity while maintaining the function's balance. We now introduce a modification of that hill climbing procedure which attempts to improve the nonlinearity and maintain balance while at the same time pushing the function towards correlation immunity. Firstly we briely review the original hill climbing algorithm.

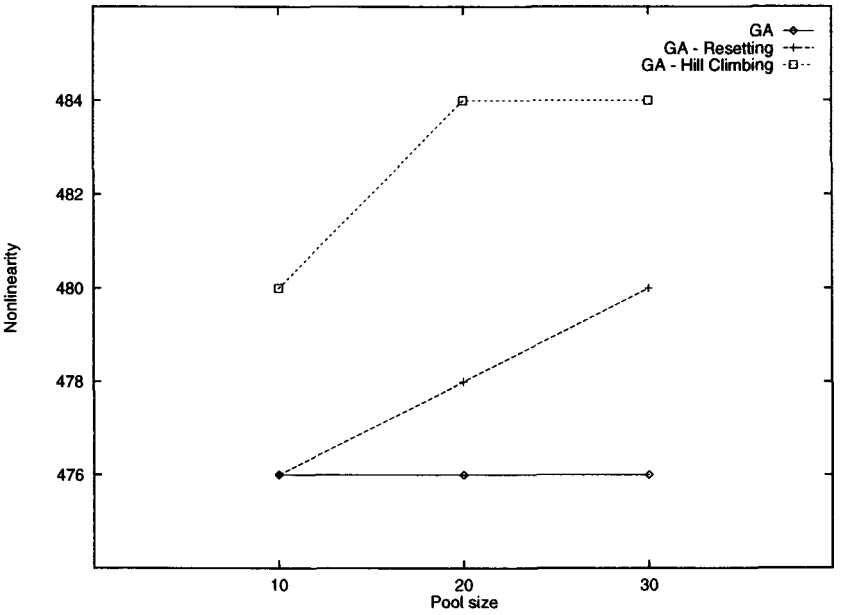

Fig. 1. Comparison (based on nonlinearity) of the different GAs for different pool sizes.

Theorem 2 in [13] defines the following five conditions which must be satisfied for the nonlinearity to increase when the truth table positions corresponding to functions inputs x_1 and x_2 are complemented:

1. $f(x_1) \neq f(x_2)$;
2. $f(x_i) = L_\omega(x_i)$, $i \in \{1,2\}$ for all $\omega \in W_1^+$;
3. $f(x_i) \neq L_\omega(x_i)$, $i \in \{1,2\}$ for all $\omega \in W_1^-$;
4. for all $\omega \in W_{2,3}^+$, if $L_\omega(x_1) \neq L_\omega(x_2)$ then $f(x_i) = L_\omega(x_i)$, $i \in \{1,2\}$; and
5. for all $\omega \in W_{2,3}^-$, if $L_\omega(x_1) \neq L_\omega(x_2)$ then $f(x_i) \neq L_\omega(x_i)$, $i \in \{1,2\}$.

The sets W are defined to be the sets of ω that specify linear functions that have minimum or near minimum Hamming distance to the Boolean function $f(x)$:

1. $W_1^+ = \{\omega : \hat{F}(\omega) = WH_{max}\}$;
2. $W_1^- = \{\omega : \hat{F}(\omega) = -WH_{max}\}$;
3. $W_2^+ = \{\omega : \hat{F}(\omega) = WH_{max} - 2\}$;
4. $W_2^- = \{\omega : \hat{F}(\omega) = -(WH_{max} - 2)\}$;
5. $W_3^+ = \{\omega : \hat{F}(\omega) = WH_{max} - 4\}$;
6. $W_3^- = \{\omega : \hat{F}(\omega) = -(WH_{max} - 4)\}$;
7. $W_{2,3}^+ = W_2^+ \bigcup W_3^+$; and
8. $W_{2,3}^- = W_2^- \bigcup W_3^-$.

By making a simple modification the hill climbing procedure will also tend towards $CI(m)$ functions. The modified algorithm assigns ω with weight less than or equal to m to the W sets in a slightly different way: the reference value is $cidev(m)$ rather than WH_{max}. This has the effect of requiring that $|\hat{F}(\omega)|$ is reduced for $|\omega| \leq m$, thus causing the algorithm to reduce $cidev(m)$ as well as increasing the nonlinearity.

This modification was found to be quite effective in finding Boolean functions which are highly nonlinear, balanced and satisfy $CI(1)$. The method has also produced balanced functions with high nonlinearity and low deviation from $CI(2)$.

4 Experimental Results

In this section we present some of the experimental results obtained. These results were obtained for small pool sizes (typically a maximum of 30). Our experiments with larger pool sizes (50, 75 and 100) did not obtain examples of functions with higher nonlinearity. The Genetic Algorithm is clearly most efficient for pools of size 30 or less.

In Table 2 the best nonlinearities obtained by some GAs seeking balanced functions are compared. The four results columns in this table represent different configurations of the genetic algorithm described above with and without resetting and hill climbing. It seems that resetting improves the performance of a GA without hill climbing. It can also be seen that the algorithms which incorporate hill climbing perform better than those which do not. Resetting was found to provide no improvement in algorithms which incorporate hill climbing.

Table 3 shows the results of a GA seeking balanced correlation immune functions. The table shows the nonlinearity and the deviation from $CI(1)$ for some of the best functions so far obtained by the hybrid genetic/hill climbing algorithm. In this case all of the best functions found were first order correlation immune. For each n the best nonlinearity obtained is close to the upper bound for balanced functions and hence must also be close to the upper bound for balanced correlation immune functions.

Similar results are presented in Table 4 for a hybrid GA seeking functions with low deviation to $CI(2)$. These functions, although not second order correlation immune, have only a small deviation from $CI(2)$ as well as high nonlinearity.

Table 5 shows typical values for the number of functions that were required to be tested in a basic GA, for $n = 8$, with various pool sizes and per bit mutation rates, in order to obtain a function with $normdev(1, 1)$ value 4 or less. This value, which we call the benchmark, is the lowest normalised deviation found in the search of one million randomly generated balanced Boolean functions. Three runs for each combination of parameters was performed from which the central value was chosen for the table data. A - indicates that the GA did not usually generate a benchmark function before converging. Some remarks can be made from the table data. Firstly small pools are more efficient in finding good functions, however if the pool is too small the GA may converge too quickly. A

	Best Nonlinearity			
n	GA	GA-Reset	GA-HC	GA-HC-Reset
8	112	114	116	116
9	234	234	236	236
10	476	480	484	484
11	970	974	980	980
12	1970	1970	1976	1976

Table 2. Comparisons of best nonlinearities for GA's with and without hill climbing and resetting.

n	Nonlinearity	$cidev(1)$
8	112	0
9	232	0
10	476	0
11	976	0
12	1972	0

Table 3. Best nonlinearity of balanced correlation immune - $CI(1)$ functions.

n	Nonlinearity	$cidev(2)$
8	112	4
9	232	8
10	480	8
11	976	8
12	1972	8

Table 4. Best nonlinearity of balanced Boolean functions and their deviation from $CI(2)$.

small amount of mutation appears to be useful in avoiding premature convergence when the pool is small, however the benefit for large pools is less clear. It is clear that too much mutation reduces the effectiveness of the algorithm. The mutation values chosen correspond roughly to none, occasional, one truth table place, and 10 places, when $n = 8$. From the table we infer that a pool size of 5 or 10, with occasional mutation, offers a near optimum compromise in the quest for benchmark functions.

n=8	Mutation Rate			
Pool	Zero	0.0004	0.004	0.04
5	-	129	146	602
10	463	292	419	828
20	987	1802	1762	1542
30	3949	2153	2487	5393

Table 5. Finding low $CI(1)/PC(1)$ deviation functions by GA, number of functions tested to get benchmark quality, typical results.

5 Conclusion

Several heuristic approaches to the design of cryptographically strong Boolean functions have been presented. These quasi-random techniques provide a suitable alternative to systematic methods for the construction of cryptographically strong Boolean functions. The concept of deviation from strict properties has been introduced and been used as a fitness function in a genetic algorithm.

Although the basic genetic algorithm is able to produce highly nonlinear balanced Boolean functions satisfying other selective properties, it is clear that several modifications improve the performance. Resetting, hill climbing and occasional mutation have proven to be effective means of improving the performance of the genetic algorithm.

References

1. E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Advances in Cryptology - Crypto '90, Proceedings, LNCS*, volume 537, pages 2–21. Springer-Verlag, 1991.
2. C. Carlet. Partially-Bent Functions. In *Advances in Cryptology - Crypto '92, Proceedings, LNCS*, volume 740, pages 280–291. Springer-Verlag, 1993.
3. A. Clark and E. Dawson. A Parallel Genetic Algorithm for Cryptanalysis of the Polyalphabetic Substitution Cipher. *Cryptologia*, 21(2):129–138, April 1997.
4. A. Clark, E. Dawson, and H. Bergen. Combinatorial Optimisation and the Knapsack Cipher. *Cryptologia*, 20(1):85–93, January 1996.
5. E. Dawson and A. Clark. Discrete Optimisation: A Powerful Tool for Cryptanalysis? In *Proceedings of Pragocrypt '96*, pages 425–451, 1996.
6. H. Dobbertin. Construction of Bent Functions and Balanced Boolean Functions with High Nonlinearity. In *Fast Software Encryption, 1994 Leuven Workshop, LNCS*, volume 1008, pages 61–74. Springer-Verlag, 1994.
7. T. Honda, T. Satoh, T. Iwata, and K. Kurosawa. Balanced Boolean functions satisfying $PC(2)$ and very large degree. In *Workshop on Selected Areas in Cryptology 1997, Workshop Record*, pages 64–72, 1997.
8. X.-D. Hou. On the Norm and Covering Radius of the First-Order Reed-Muller Codes. *IEEE Transactions on Information Theory*, 43(3):1025–1027, May 1997.
9. F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error Correcting Codes.* North-Holland Publishing Company, Amsterdam, 1978.

10. M. Matsui. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology - Eurocrypt '93, Proceedings, LNCS*, volume 765, pages 386–397. Springer-Verlag, 1994.
11. Robert A. J. Matthews. The use of genetic algorithms in cryptanalysis. *Cryptologia*, 17(2):187–201, April 1993.
12. W. Millan, A. Clark, and E. Dawson. An Effective Genetic Algorithm for Finding Highly Nonlinear Boolean Functions. In *First International Conference on Information and Communications Security*, volume 1334 of *Lecture Notes in Computer Science*, pages 149–158. Springer-Verlag, 1997.
13. W. Millan, A. Clark, and E. Dawson. Smart Hill Climbing Finds Better Boolean Functions. In *Workshop on Selected Areas in Cryptology 1997, Workshop Record*, pages 50–63, 1997.
14. N.J. Patterson and D.H. Wiedemann. The Covering Radius of the $(2^{15}, 16)$ Reed-Muller Code is at least 16276. *IEEE Transactions on Information Theory*, 29(3):354–356, May 1983.
15. B. Preneel. *Analysis and Design of Cryptographic Hash Functions.* PhD thesis, Cathoic University of Leuven, 1994.
16. B. Preneel, W. Van Leekwijck, L. Van Linden, R. Govaerts, and J. Vandewalle. Propagation Characteristics of Boolean Functions. In *Advances in Cryptology - Eurocrypt '90, Proceedings, LNCS*, volume 473, pages 161–173. Springer-Verlag, 1991.
17. M. Schneider. On the Construction and Upper Bounds of Balanced and Correlation-immune Functions. In *Workshop on Selected Areas in Cryptology 1997, Workshop Record*, pages 73–87, 1997.
18. J. Seberry, X.-M. Zhang, and Y. Zheng. Nonlinearly balanced boolean functions and their propagation characteristics. In *Advances in Cryptology - Crypto '93, Proceedings, LNCS*, volume 773, pages 49–60. Springer-Verlag, 1994.
19. J. Seberry, X.-M. Zhang, and Y. Zheng. On Constructions and Nonlinearity of Correlation Immune Functions. In *Advances in Cryptology - Eurocrypt '93, Proceedings, LNCS*, pages 181–199. Springer-Verlag, 1994.
20. T. Siegenthaler. Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications. *IEEE Transactions on Information Theory*, 30(5):776–780, September 1984.
21. J.J. Son, J.I. Lim, S. Chee, and S.H. Sung. Global Avalanche Characteristics and Nonlinearity of Balanced Boolean Functions, 1997. Submtted to Information Processing Letters.
22. R. Spillman. Cryptanalysis of Knapsack Ciphers using Genetic Algorithms. *Cryptologia*, 17(4):367–377, October 1993.
23. R. Spillman, M. Janssen, B. Nelson, and M. Kepner. Use of a Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers. *Cryptologia*, 17(1):31–44, January 1993.
24. D.R. Stinson. Resilient functions and large sets of orthogonal arrays. *Congressus Numerantium*, 92:105–110, 1993.
25. A.F. Webster and S.E. Tavares. On the Design of S-Boxes. In *Advances in Cryptology - Crypto '85, Proceedings, LNCS*, volume 218, pages 523–534. Springer-Verlag, 1986.
26. G-Z. Xiao and J.L. Massey. A Spectral Characterization of Correlation-Immune Combining Functions. *IEEE Transactions on Information Theory*, 34(3):569–571, May 1988.