Non-linear Cryptanalysis Revisited: Heuristic Search for Approximations to S-Boxes

Juan M.E. Tapiador[1], John A. Clark[1], and Julio C. Hernandez-Castro[2]

[1] Department of Computer Science, University of York
York YO10 5DD, England, UK
{jet, jac}@cs.york.ac.uk

[2] Department of Computer Science, Carlos III University of Madrid
28911 Leganes (Madrid), Spain
jcesar@inf.uc3m.es

Abstract. Non-linear cryptanalysis is a natural extension to Matsui's linear cryptanalitic techniques in which linear approximations are replaced by non-linear expressions. Non-linear approximations often exhibit greater absolute biases than linear ones, so it would appear that more powerful attacks may be mounted. However, their use presents two main drawbacks. The first is that in the general case no joint approximation can be done for more than one round of a block cipher. Despite this limitation, Knudsen and Robshaw showed that they can be still very useful, for they allow the cryptanalist greater flexibility in mounting a classic linear cryptanalysis. The second problem concerning non-linear functions is how to identify them efficiently, given that the search space is superexponential in the number of variables. As the size of S-boxes (the elements usually approximated) increases, the computational resources available to the cryptanalyst for the search become rapidly insufficient.

In this work, we tackle this last problem by using heuristic search techniques –particularly Simulated Annealing– along with a specific representation strategy that greatly facilitates the identification. We illustrate our approach with the 9×32 S-box of the MARS block cipher. For it, we have found multiple approximations with biases considerably larger (e.g. $151/512$) than the best known linear mask ($84/512$) in reasonable time. Finally, an analysis concerning the search dynamics and its effectiveness is also provided.

1 Introduction

The adoption of the Data Encryption Standard (DES) [26,27] provided an extraordinary stimulus for the development of public cryptology, and particularly for the advancement of modern cryptanalytic methods. In the context of DES analysis, Reeds and Manferdelli [28] introduced in 1984 the idea of "partial linearity", pointing out that a block cipher with such a characteristic may be vulnerable to known- or chosen-plaintext attacks faster than exhaustive key search. Chaum and Evertse subsequently extended the notion of a per round linear factor to that of "sequence of linear factors" [5], proving that DES versions with more than 5 rounds had no partial linearity caused by such a sequence. These concepts were later generalized to that of "linear structure" [8], which embraced properties such as the complementarity of DES or the existence of bit independencies (i.e. some bits of the ciphertext being independent of the values of certain plaintext and key bits), among others.

In a sense, linear cryptanalysis constituted the natural extension to these efforts. Linear cryptanalysis was introduced by Matsui in [17] as a potential technique to attack DES. Its applicability was corroborated soon after [18], in what is commonly accepted as the first –although barely practical at the time– compromise of the cipher.

Several refinements to the basic idea of linear cryptanalysis have attempted to improve the efficiency of the attacks, either in specific circumstances or in all cases. As soon as 1994, Kaliski and Robshaw proposed an extension based on the use of multiple linear approximations [13]. Harpes, Kramer and Massey [11] presented in 1995 a generalisation in which linear expressions are replaced by I/O sums. An I/O sum is the XOR of a balanced binary-valued function of the input and a balanced binary-valued function of the output.

Beyond these improvements, the most natural idea is to consider whether the linear approximations can be replaced with non-linear approximations. In 1996, Knudsen and Robshaw introduced the idea of extending Matsui's linear cryptanalytic techniques to the more general case in which non-linear relations are also considered [15]. To motivate this approach, they provide a practical example showing that it is feasible to obtain much more accurate approximations to DES S-boxes by considering non-linear relations instead of linear ones. In that same work, they identified non-linear approximations to the S-boxes of LOKI91 [2], a DES-like block cipher that operates on 64-bit blocks and uses a 64-bit key. One of the most remarkable features of LOKI91 is that it uses four identical S-boxes which map 12 to 8 bits. While the three best linear approximations known to these S-boxes exhibited biases of 88/4096, 108/4096 and 116/4096, the authors found non-linear relations with biases of 136/4096, 130/4096 and 110/4096.

1.1 Motivation and Related Work

Linear cryptanalysis was proposed to attack block ciphers and mainly applied to Feistel-like constructions [9]. In this type of design, the overall effect of a round is entirely linear (and often independent of the key) except for a single component, which is typically implemented using one or more S-boxes. It is precisely in this context wherein the search for (linear) approximations of these components makes sense.

As S-boxes (especially in the past) were often fairly small, the search space in which to look for linear approximations was small enough, in many cases allowing an exhaustive search in a reasonable amount of time. As a result, the method for determining the best linear approximation has not been itself a matter of extensive study (an early exception to this was [19]). However, the situation becomes dramatically different when considering non-linear approximations: there are 2^{2^n} different Boolean functions of n variables (recall that only 2^n are linear). Even for a low number of inputs (e.g. $n = 8$) the search space is astronomically huge, so a brute-force approach will simply not work.

There is, however, a different but very related field that has been tackled quite successfully by applying heuristic search techniques: the design and analysis of cryptographic Boolean functions.

The design of Boolean functions with desirable cryptographic properties (e.g. high non-linearity, low autocorrelation, high algebraic degree, reasonable order of correlation immunity, etc.) has traditionally been a central area of cryptological research. In the latter half of the 1990s, a few works suggested that heuristic search techniques could be applied to efficiently derive good Boolean functions. Millan et al. [20] were the first to show that small changes to the truth table of Boolean functions do not radically alter their non-linearity nor their autocorrelation properties. This provides for an efficient delta-fitness function for local searches. Later works by the same authors (e.g. [21]) demonstrated that Boolean functions that are correlation-immune or satisfying the strict avalanche criterion can be found too with smart hill-climbing.

This approach was subsequently generalized in a number of ways. Millan et al. applied it to the design of bijective [22] and regular [23] S-boxes. Clark and Jacob used Simulated Annealing in [6] to achieve some results hitherto unattained by other means. Some of the results presented in that work constituted also a counter-example to a then existing conjecture on autocorrelation.

1.2 Contribution and Overview

The main purpose of this paper is to show that heuristic methods very similar to that used for designing cryptographic Boolean functions can be applied to find non-linear approximations to S-boxes. In Section 2 we introduce some basic concepts on non-linear cryptanalysis, with particular emphasis in the elements which are essentially different to linear cryptanalysis. We describe our approach and the experimental setup used in Section 3. The most relevant results of the experiments carried out are shown in Section 4, together with an analysis concerning the efficiency and statistical significance of the search. Finally, in Section 5 we make a few concluding remarks and outline some possible extensions to this work.

2 Basic Concepts of Non-linear Cryptanalysis

Consider an n variable Boolean function $f : GF(2^n) \rightarrow GF(2)$. The (binary) truth table (TT) of f is a vector of 2^n elements representing the output of the function for each input. Each Boolean function has a unique representation in the Algebraic Normal Form (ANF) as sum of product terms:

$$\begin{aligned} f(x_1, \ldots, x_n) = {} & a_0 \oplus a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n \\ & a_{12} x_1 x_2 \oplus \cdots \; a_{n-1,n} x_{n-1} x_n \\ & \oplus \cdots \oplus a_{1,\ldots,n} x_1 x_2 \cdots x_n \end{aligned} \tag{1}$$

The order of each product term in the ANF is defined as the number of variables the product term contains. The *algebraic order* of f, denoted $ord(f)$, is the maximum order of the product terms in the ANF for which the coefficient is 1.

There are other widely-known representations of Boolean functions, such as the polarity truth table or the Walsh-Hadamard spectrum, which nonetheless shall not be used in this work.

The *Hamming weight* of a Boolean function f of n variables, denoted by $hw(f)$, is the number of ones in its TT:

$$hw(f) = \sum_{x=0}^{2^n-1} f(x) \tag{2}$$

A function f is said to be *balanced* iff $hw(f) = 2^{n-1}$.

Let $x = (x_1, x_2, \ldots, x_n)$ and $\omega = (\omega_1, \omega_2, \ldots, \omega_n)$ be binary n-tuples, and:

$$\omega \cdot x = \bigoplus_{i=1}^{n} \omega_i x_i \tag{3}$$

their dot product. Then the linear function $L_\omega(x)$ is defined as $L_\omega(x) = \omega \cdot x$. The set of affine functions consists of the set of linear functions and their complements. Note that every linear function is balanced.

A $n \times m$ substitution box (or simply S-box) is a mapping from n input bits to m output bits. It can also be viewed as an ordered set of m Boolean functions of n variables each.

2.1 Non-linear Approximations to S-Boxes

In the following definitions S represents a $n \times m$ S-box, $x = (x_1, \ldots, x_n)$ the input to S, and $y = (y_1, \ldots, y_m)$ the corresponding output. For compatibility with the notation often used in the literature on linear cryptanalysis, we shall write the inner product $\omega \cdot x$ of two binary n-tuples ω and x, as $x[\omega]$. We shall keep this notation even when approximations are not linear, i.e. if f is a non-linear function, then $x[f] = f(x)$. An exception to this is the output produced by S-boxes, which always shall be written as $S(x)$.

Definition 1. *A non-linear approximation for S is a pair*

$$A = \langle \Gamma_x, \Gamma_y \rangle \tag{4}$$

where $\Gamma_x : GF(2^n) \rightarrow GF(2)$ and $\Gamma_y : GF(2^m) \rightarrow GF(2)$.

It should be clear that, even though the domains of Γ_x and Γ_y are, respectively, $GF(2^n)$ and $GF(2^m)$, in order for them to be cryptanalytically useful they must not depend on all the variables in x and y. This implies that it must be possible to write them as functions in $GF(2^{\hat{n}})$ and $GF(2^{\hat{m}})$, respectively, with $\hat{n} < n$ and $\hat{m} < m$.

Definition 2. *Let $A = \langle \Gamma_x, \Gamma_y \rangle$ be a non-linear approximation and $P_S(A)$ the probability that the relation:*

$$x[\Gamma_x] \oplus S(x)[\Gamma_y] = 0 \tag{5}$$

holds for an S-box S. Let $P_R(A)$ be the probability that the same relation holds for a random $n \times m$ bit permutation R. The deviation of A with respect to S, denoted by $\delta_S(A)$, is given by:

$$\delta_S(A) = P_S(A) - P_R(A) \tag{6}$$

Definition 3. *The absolute value of the deviation is called the bias of A with respect to S:*

$$\epsilon_S(A) = |\delta_S(A)| \tag{7}$$

When the context will be clear enough, the subscript S will be omitted for denoting the deviation and bias.

2.2 Computing the Bias of a Non-linear Approximation

The computation of $P_S(A)$ can be done by exploring the 2^n possible inputs to $S(x)$ and checking how many times expression (5) holds. In cases where n is large, a random sampling over the inputs will provide an estimation of $P_S(A)$.

On the other hand, the computation of $P_R(A)$ depends on the Hamming weight of the functions integrating the approximation. Under the assumption of randomness of x, we have that:

$$Px(0) = \text{Prob}\big(x[\mathit{\Gamma}_x] = 0\big) = 1 - \frac{1}{2^n} hw(\mathit{\Gamma}_x) \tag{8}$$

$$Px(1) = \text{Prob}\big(x[\mathit{\Gamma}_x] = 1\big) = \frac{1}{2^n} hw(\mathit{\Gamma}_x) \tag{9}$$

$$Py(0) = \text{Prob}\big(R(x)[\mathit{\Gamma}_y] = 0\big) = 1 - \frac{1}{2^m} hw(\mathit{\Gamma}_y) \tag{10}$$

$$Py(1) = \text{Prob}\big(R(x)[\mathit{\Gamma}_y] = 1\big) = \frac{1}{2^m} hw(\mathit{\Gamma}_y) \tag{11}$$

Expression (5) can be now computed by simply applying the definition of the sum over GF(2):

$$P_R(A) = Px(0) \cdot Py(1) + Px(1) \cdot Py(0) \tag{12}$$

Substituting and regrouping terms we have:

$$P_R(A) = \frac{1}{2^n} hw(\mathit{\Gamma}_x) + \frac{1}{2^m} hw(\mathit{\Gamma}_y) - \frac{2}{2^{n+m}} hw(\mathit{\Gamma}_x) \cdot hw(\mathit{\Gamma}_y) \tag{13}$$

A particularly interesting observation is that if at least one of the two functions (e.g. $\mathit{\Gamma}_x$) is balanced, then:

$$\begin{aligned} P_R(A) &= \frac{1}{2^n} hw(\mathit{\Gamma}_x) + \frac{1}{2^m} hw(\mathit{\Gamma}_y) - \frac{2}{2^{n+m}} hw(\mathit{\Gamma}_x) \cdot hw(\mathit{\Gamma}_y) \\ &= \frac{1}{2^n} \frac{2^n}{2} + \frac{1}{2^m} hw(\mathit{\Gamma}_y) - \frac{2}{2^{n+m}} \frac{2^n}{2} \cdot hw(\mathit{\Gamma}_y) = \frac{1}{2} \end{aligned} \tag{14}$$

The same is applicable in case of $\mathit{\Gamma}_y$ being balanced. Note that this is precisely the case when one of the two functions is linear. In these cases, calculating the bias is considerably more efficient.

2.3 Joint Approximations, Limitations and Cryptanalytic Utility

In order to make a practical use of a non-linear approximation to an S-box, one needs to extend it into an approximation across the entire round function. This task is usually strongly dependant on the particular structure of the round function used and the way its output is subsequently processed through the rest of operations comprising a round.

In nearly all cases, a common problem is how to relate the input to the S-box with the actual inputs to the round function. The usual operation of a round function consists in first combining certain key bits (or a subkey) with some bits from the input data block, and then apply the S-box to the result. Suppose that d and k are the input data block and key to a round function, respectively, and that both are of the same length. Assume that the round function operates by applying an S-box to $x = d \oplus k$. In case of L being a linear approximation, the dependency on the key can be easily peeled off, since:

$$x[L] = (d \oplus k)[L] = d[L] \oplus k[L] \tag{15}$$

However, it is not generally possible to do this for a non-linear approximation. Furthermore, the actual approximation applied depends on the specific values of d and k. A detailed analysis of this phenomenon along with specific examples of how to approximate a round function can be found in [15] and [25].

Consider now a Feistel cipher [9] where the input data to round i is denoted as C_h^{i-1} and C_l^{i-1} (the high-order and low-order halves of the data block, respectively). The action of the round function will be denoted by $f(C_l^{i-1}, k_i)$, k_i being the subkey corresponding to this round. With this notation, the output from the i^{th} round is written as $C_h^i = C_l^{i-1}$ and $C_l^i = C_h^{i-1} \oplus f(C_l^{i-1}, k_i)$.

If α, β, γ and δ are linear masks specifying a selected subgroup of bits, then a *linear* approximation to a single round can be written as [15]:

$$C_h^{i-1}[\alpha] \oplus C_l^{i-1}[\beta] = C_h^i[\gamma] \oplus C_l^i[\alpha] \oplus k_i[\delta] \tag{16}$$

Rewriting previous expression as:

$$C_l^{i-1}[\beta \oplus \gamma] \oplus k_i[\delta] = C_h^{i-1}[\alpha] \oplus C_l^i[\alpha] = (C_h^{i-1} \oplus C_l^i)[\alpha] \tag{17}$$

we obtain an approximation to round i.

Assume now that $f(\alpha)$ is a non-linear function. Again, it is obvious that the relation:

$$(C_h^{i-1} \oplus C_l^i)[f(\alpha)] = C_h^{i-1}[f(\alpha)] \oplus C_l^i[f(\alpha)] \tag{18}$$

will not generally hold.

In summary, one-round approximations that are not-linear in the output bits from $f(C_l^{i-1}, k_i)$ cannot be generally joined together. However, as noted by Knudsen and Robshaw in [15], they can be useful in a number of ways. For the first and last rounds of a cipher, the input to an approximation need not to be combined with any other approximation. Therefore, non-linear approximations can be used in these rounds without concern. Moreover, both the 1R- and 2R-methods of linear cryptanalysis proposed by Matsui (see [17]) require that some bits of the input to the second (or penultimate) round will be available to the cryptanalyst. By using this fact, non-linear approximations can be used in these rounds too.

The previous applications of non-linear approximations may allow the cryptanalyst to recover, in some circumstances, more key bits less plaintexts than required by linear techniques. As an example, in [15] it is shown how a non-linear cryptanalysis of reduced-round LOKI91 [2] allows to recover 7 additional key bits with less than 1/4 of the plaintexts required by linear techniques.

3 Heuristic Search for Non-linear Approximations

We wish to explore the space of possible approximations by searching for a solution that maximises the bias as defined by expression (7). The search space has a size of $\mathcal{O}(2^{2^n+2^m})$, n and m being the input and output sizes of the S-box, respectively. Obviously, this makes exhaustive search impossible (by standard computation means, at least) even for relatively low values of n and m.

In the experiments reported in this paper, we have used the well-established technique of Simulated Annealing [14]. A description of its operation is provided in Appendix B. In order to apply it to our problem, we have to provide three basic elements: a descriptive characterisation of the search space (i.e. a representation for solutions), a move function defining a neighbourhood for each element in the search space, and a fitness (or cost) function measuring how good a potential solution is. Next we describe these three components.

3.1 Solutions

The representation and operations for evolving Boolean functions used in this work are based on those provided by Fuller, Millan and Dawson in [10]. Each candidate approximation A is a structure comprising functions Γ_x and Γ_y, as given by Definition 1. Function Γ_x is represented by three elements:

– The number $\hat{n} < n$ of variables of the approximation.
– The TT of the non-linear approximation (of $\hat{n}$ variables).
– An injective function $J : \mathbb{Z}_{\hat{n}} \rightarrow \mathbb{Z}_n$ assigning each variable in the approximation to one input variable of the S-box. We shall represent J by a vector with its values: $(J(0), J(1), \ldots, J(\hat{n}-1))$. The terms $\hat{n}$ and $\dim(J)$ shall be used interchangeably.

The purpose of J is to *project* the non-linear function into some of the input bits of the S-box. This is necessary due to the reasons previously discussed –only approximations making use of some input and output bits are cryptanalytically useful.

The next example illustrates how this structure should be interpreted. Suppose S is a 4×8 S-box, $\hat{n} = 3$, and an approximation for the input variables is given by:

$$\Gamma(z_0, z_1, z_2) = z_0 \oplus z_0 z_1 \oplus z_0 z_1 z_2$$

Assuming $J_1 = (1, 3, 0)$, the projection of Γ into the input bits of S is given by:

$$\Gamma_x^{J_1}(x_0, x_1, x_2, x_3) = x_1 \oplus x_1 x_3 \oplus x_1 x_3 x_0$$

while for $J_2 = (0, 2, 1)$ it would be:

$$\Gamma_x^{J_2}(x_0, x_1, x_2, x_3) = x_0 \oplus x_0 x_2 \oplus x_0 x_2 x_1$$

Note that, given an input approximation and fixed values for n and $\hat{n}$, the number of different projections is exactly the number of permutations:

$$P_{\hat{n}}^{n} = \frac{n!}{(n - \hat{n})!} \tag{19}$$

As it will be discussed later, this value should be taken into account when defining some of the search parameters.

A similar representation could be provided to Γ_y. In our experiments, however, we decided to let it simply be a linear mask rather than a general non-linear function. This presents some remarkable advantages. From a complexity standpoint, it reduces the amount of space required to represent a solution, reduces considerably the search space, and also makes the search much faster (recall that linearity in one function implies a much simpler way of computing the bias of the whole approximation). Moreover, S-boxes are non-linear in their input bits, so it seems reasonable to look for approximations with non-linearity just in them, even though non-linear combinations of output bits might have some cryptanalytic interest too.

3.2 Move Function

To obtain a candidate $A' \in N(A)$ in the neighbourhood of A, we have defined a move function governed by five parameters:

- The values P_x and P_y, with:

$$\begin{aligned} 0 \leq P_x \leq P_y &\leq 1 \\ P_x + P_y &\leq 1 \end{aligned}$$

  which define if a neighbour of A will be obtained by changing Γ_x, Γ_y or J, and keeping the other two unaltered. Specifically:
  - P_x is the probability of moving Γ_x,
  - $P_y - P_x$ is the probability of moving Γ_y, and
  - $1 - P_y$ is the probability of moving J.
- The parameters C_x, C_y and C_J control how many elements of Γ_x, Γ_y or J, respectively, will be mutated. In case of Γ_x and Γ_y, this is the number of bits to be flipped, while for J it defines the number of elements in the projection to be substituted by a new one. This new component is randomly generated and, as J is an injective function, must be different to each element already present in J.

The basic operation of the move function is described in Fig. 1.

3.3 Evaluation and Fitness

The fitness function used to guide the search is simply the bias achieved by the approximation:

$$F(A) = \epsilon \tag{20}$$

as defined by expression (7). This is computed by generating the 2^n different input values to the S-box. Each value is then translated to the corresponding projected value according to J, and the output value produced by Γ_x is stored. For the same (not projected) input value, the actual output produced by the S-box is computed and the mask Γ_y is applied. The resulting value is stored together with the previous one.

```
Pick $u \in [0,1]$ with uniform probability
if $0 \le u \le P_x$ then
    for $i = 1$ to $C_x$
        Pick $r \in [0, 2^{\hat{n}} - 1]$ with uniform probability
        Flip bit $r$ in the TT of $\Gamma_x$
else if $P_x < u \le P_y$ then
    for $i = 1$ to $C_y$
        Pick $r \in [0, m-1]$ with uniform probability
        Flip bit $r$ in the TT of $\Gamma_y$
else if $P_y < u \le 1$ then
    for $i = 1$ to $C_J$
        Pick $r \in [0, \hat{n} - 1]$ with uniform probability
        Pick $v \in [0, n-1]$ with uniform probability, and such that $v \notin J$
        $J(r) \leftarrow v$
```

Fig. 1. Move function for evolving functions Γ_x and Γ_y, and projection J

Upon reaching the end of the process, the bias can be calculated as:

$$\epsilon = \left| \frac{E}{2^n} - \frac{1}{2} \right| \tag{21}$$

E being the number of times both outputs coincided. For subsequent analysis, it is also useful to define this magnitude in absolute terms:

$$Hits = |E - 2^{n-1}| \tag{22}$$

so the bias can be represented as $Hits/2^n$ (recall the maximum value for $Hits$ is 2^{n-1}). Here it is important to note that, under the assumption of random input, the expected hit rate is 0 due to the linearity of Γ_y. In a general case where Γ_x and Γ_y are both non-linear, the expected hit rate has to be computed as described in Section 2.2.

4 Experimental Results and Analysis

We have applied the technique briefly described above to the S-box included in the block cipher MARS [4]. MARS was IBM's candidate submission to the AES contest and, as such, is nowadays a quite well-studied scheme.

One of the core components of MARS is a 9×32 S-box with very specific combinatorial, differential and linear correlation properties. This S-box was the outcome of a search process that took IBM about a week of intensive computation. Burnett et al. showed in [3] that 9×32 S-boxes with cryptographic properties clearly superior to those of MARS S-box can be found with heuristic search methods in less than 2 hours on a single PC. In 2000, Robshaw and Yin [29] made some comments about the resistance of this S-box to linear cryptanalysis, challenging the claims by IBM's designers. Soon after, Knudsen and Raddum [16] found a large number of linear approximations with biases higher than 2^{-3}, again contradicting a conjecture made by the designers. Among the 871 linear masks found in that work, the best exhibits a bias of 82/512. Soon after, Aoki was able to compute the complete distribution of MARS's S-box linear masks [1], finding that the best linear approximation had a bias of 84/512.

In response to these criticisms, the MARS team issued some comments (see e.g. [12]) pointing out that some of their claims were poorly worded and hence easily misinterpreted. Basically, they argued that it is the MARS *core* function which actually has no bias higher than a given magnitude (2^{-3}), but not specific components such as, for example, the S-box itself.

One of the reasons for choosing MARS S-box to illustrate our approach is (apart from the fact that it was one of the five AES finalists) precisely the existence of linear masks with such a large bias. From this point of view, finding non-linear approximations with biases much larger than those –and in considerably less time– is an interesting challenge.

4.1 Parameterisation

In our preliminary experimentation, we found that the search dynamics is quite sensitive to the values $(P_x, P_y, C_x, C_y, C_j)$ required by the move function. For the last three, the best results are systematically found when the number of changes is low. This seems reasonable, since a strong mutation of a candidate implies an abrupt move in the search space, often resulting in a loss of many of the properties obtained so far.

Concerning the probabilities of mutation assigned to each component, the probability of moving J should be somehow related to the number of different projections as defined by expression (19). If the number of possible projections is large and the associated probability low, the search will be mostly devoted to find functions that "fits" a projection which is rarely changed. Alternatively, when the number of projections is relatively low, a high probability will result in an inefficient search eventually repeating candidate solutions.

In our experiments, the best results were obtained by using the parameterisation shown in Table 1. The probability of moving J is decreased as the number of possible permutations gets lower. A probability higher than 0.5 did not demonstrate better performance.

Each inner loop tries 10000 moves, and the number of inner loops is bounded by 1000. The search stops whenever this number is reached or after 250 consecutive inner loops without improvement.

4.2 Results

We ran 25 experiments for each value of $\hat{n}$ from 2 to 8. Each experiment takes around 1 hour in a PC (Intel Pentium 4 CPU 2.80 GHz with 1 GB RAM.)

For values of $\hat{n}$ from 2 to 4, no approximation better than the best known linear mask was found. In these cases, the functions found exhibit biases between 60 and 75. This is also the case for $\hat{n} = 5$, even though eventually we found two approximations with bias 83/512, i.e. very similar to the best linear mask.

In the case of approximations using 6, 7 or 8 out of the 9 input bits, the results were considerably better, achieving approximations with an average bias of around 89, 110

Table 1. Simulated Annealing parameters

GENERAL	
Max. No. inner loops	1000
Max. No. moves in inner loop	10000
Max. No. failed inner loops	250
Initial temperature	200
Cooling rate	0.99

EVALUATION: MARS S-box	
n	9 bits
m	32 bits

MOVE FUNCTION
$\hat{n} = 8 \rightarrow (P_x, P_y, C_x, C_y, C_J) = (0.25, 0.25, 1, 1, 1)$
$\hat{n} = 7 \rightarrow (P_x, P_y, C_x, C_y, C_J) = (0.30, 0.30, 1, 1, 1)$
$\hat{n} = 6 \rightarrow (P_x, P_y, C_x, C_y, C_J) = (0.35, 0.35, 1, 1, 1)$
$\hat{n} = 5 \rightarrow (P_x, P_y, C_x, C_y, C_J) = (0.40, 0.40, 1, 1, 1)$
$\hat{n} = 4 \rightarrow (P_x, P_y, C_x, C_y, C_J) = (0.45, 0.45, 1, 1, 1)$
$\hat{n} = 3 \rightarrow (P_x, P_y, C_x, C_y, C_J) = (0.45, 0.45, 1, 1, 1)$
$\hat{n} = 2 \rightarrow (P_x, P_y, C_x, C_y, C_J) = (0.45, 0.45, 1, 1, 1)$

and 140, respectively. Fig. 2 shows the bias distribution for the 25 approximations found in each case.

The best among them constitute certainly an interesting result. For $\hat{n} = 8$, we found an approximation with bias 151/512, which translates to a deviation of around 0.29. Something similar occurs for $\hat{n} = 7$ and 6, for which the best approximations show deviations of 0.23 and 0.18, respectively. A remarkable point is the effectiveness of the search: even though finding the "best" approximation for a given number of input bits may require several runs, the search consistently provides good candidates.

Finally, the two best approximations found for 6, 7 and 8 variables are provided in Appendix A.

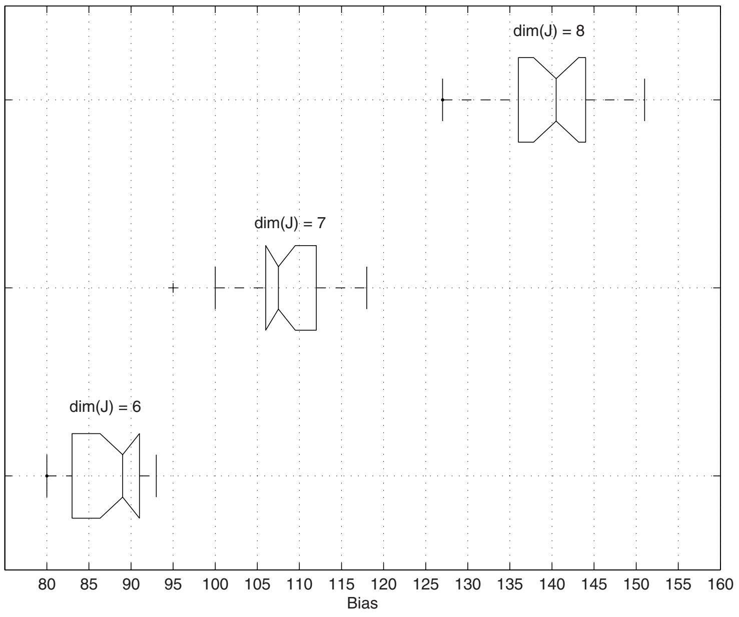
4.3 Effectiveness of the Heuristic

Now we provide a brief analysis concerning the statistical significance of our results. The purpose of this is to show that the search is indeed effective, i.e. it behaves considerably better than what should be expected from a pure blind search.

Computing the bias of a given approximation to MARS S-box can be seen as performing 512 experiments, each of which can result in a 1 (real and predicted output match) or 0 (when not). If we perform 512 independent experiments of a Binomial $B(1, 1/2)$, the result, by the additive property of the Binomial probability distribution, should behave as a $B(512, 1/2)$, whose standard deviation is $\sigma = \sqrt{0.5 \cdot 0.5 \cdot 512} = \sqrt{128}$. It is well known that in certain conditions ($n \geq 30$, $np \geq 5$ and $n(1-p) \geq 5$), a Binomial $B(n, p)$ can be accurately approximated by a $N(np, np(1-p))$. As these conditions clearly hold in our case, we can safely approximate a $B(512, 1/2)$ by a $N(256, 128)$; or, equivalently:

$$\frac{E - 256}{\sqrt{128}} = \frac{\pm Hits}{\sqrt{128}} \sim N(0, 1) \tag{23}$$

E being, as in (21), the number of times both outputs match. Even if the S-box were generated completely at random, if the number of experiments is high enough, a random search would find candidates increasingly "better." Therefore, the number of total



$\dim(J) = 6$

Bias	Number of Solutions
80	2
83	4
86	2
87	3
88	1
89	2
90	4
91	2
92	4
93	1

$\dim(J) = 7$

Bias	Number of Solutions
95	1
100	1
101	1
102	2
103	1
106	5
107	1
108	3
109	2
112	4
115	2
118	2

$\dim(J) = 8$

Bias	Number of Solutions
127	1
129	1
131	1
132	1
135	1
136	3
137	1
138	2
140	2
141	1
143	3
144	2
145	2
147	2
148	1
151	1

Fig. 2. Bias distribution for 25 experiments with $\dim(J) = 6$, 7 and 8. Each boxplot has lines at the lower quartile, median, and upper quartile values. The lines extending from each box show the extent of the rest of the data.

Table 2. Statistical significance for some bias values

Bias	Equivalent value observed in a $N(0,1)$	Probability
150/512	13.25	3.00×10^{-39}
140/512	12.37	2.36×10^{-34}
130/512	11.49	8.57×10^{-30}
120/512	10.61	1.43×10^{-25}
110/512	9.72	1.22×10^{-21}
100/512	8.84	4.28×10^{-18}
90/512	7.95	7.53×10^{-15}
80/512	7.07	5.58×10^{-12}

evaluations performed during the search process (say I) should be somehow incorporated into the analysis. For our purposes, if:

$$1 - \frac{1}{I} = \mathrm{erf}(\frac{n}{\sqrt{2}}) \tag{24}$$

where:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{25}$$

is the Gauss error function, then we should expect a number of hits at around n standard deviations from the mean.

In the case of the best approximations obtained for $\hat{n} = 8$, a bias of $151/512$ implies 151 hits, which translates to $151/\sigma \simeq 13.35$ standard deviations away from the mean. This is statistically equivalent to observe a value of around 13.35 coming from a $N(0,1)$, an extremely unusual event with an associated probability of occurrence of around 7.95×10^{-40}. This means that by following a pure blind search, the average number of evaluations required to yield this number is around 7.95×10^{40}. Recall that our search is bounded by 10^7 evaluations (actually, in most cases the stopping criterion is met at around 10^6; this shall be discussed below.)

Table 2 shows the equivalent value observed in a $N(0,1)$ and its associated probability for some values in the range of the best approximations found in our experimentation. From a simple inspection of these numbers, it should be clear that the heuristic is effectively achieving solutions that, otherwise, would not have been feasible to a random search.

4.4 Search Dynamics

Figure 3 shows the evolution of the fitness value associated with the best candidate found so far in a typical search. Around the first 500000 movements are completely random, during which the algorithm tries to locate a good candidate in the search space. In almost all the executions tried, the next behaviour has been identical: once an "appropriate" candidate is found, its fitness is considerably incremented in the next 500000 movements. This corresponds to the rapid growth observed in the curve, during which the resulting candidate is constantly refined until no more improvement can be done.

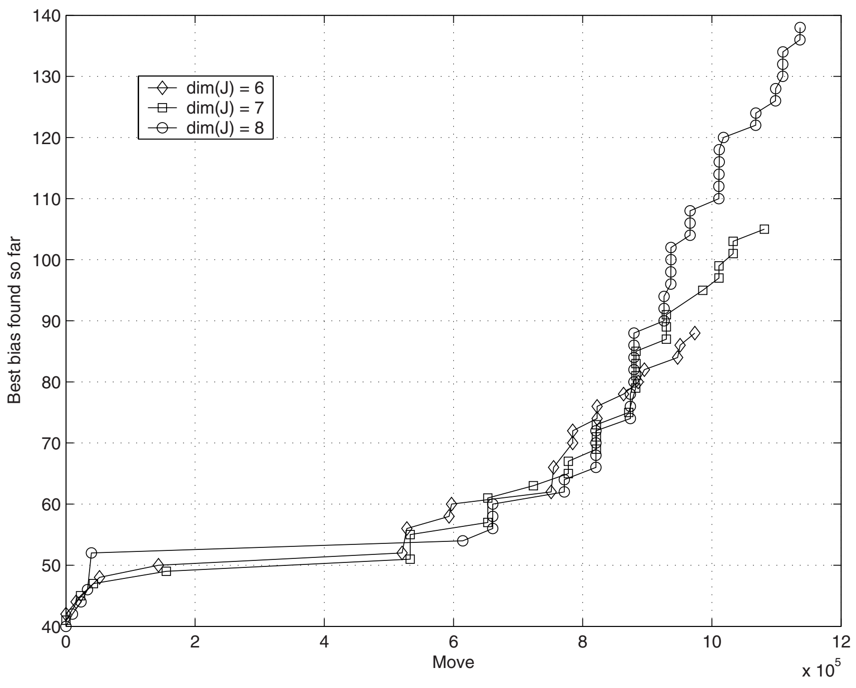


Fig. 3. Search dynamics during a typical running. The graph shows the evolution of the bias corresponding to the best approximation found so far.

In nearly all cases, the search stops improving solutions at around 1 or 1.2 millions of evaluations (i.e. 100 inner loops). After that, the next 250 cycles are completely useless and, therefore, the stop criterion is met and the search stops. This behaviour suggests that the stopping criterion can be greatly relaxed, stopping the search after 80 or 100 non-improving inner loops, instead of 250. This would result in searches of around 40 minutes of running time each, rather than the 1 hour on average pointed out previously.

4.5 Discussion

The heuristic is obviously working at improving the quality of the candidates, reaching approximations that might not have been found by a blind search. However, though effective, we are probably not using an optimal approach in the search. It is not clear to us whether the fitness function is appropriate for providing guidance, so alternative measures should be considered. Moreover, a restriction of the search space could be useful too. In a similar problem, Millan et al. [24] showed how searching over the space of ANFs of Boolean functions of degree less than or equal to $n/2$ can be used to provide Bent functions of highest possible degree. In [7], Clark et al. searched over the space of Walsh spectra for a permutation of an initial spectrum that corresponds to a Boolean function.

One of the most interesting features of both papers is that the form of representation used greatly facilitates the restriction of the possible solutions. In Millan et al.'s work, bent function theory shows that the maximal algebraic degree is $n/2$, and so restricting the space to functions with terms of or lower than this degree is clearly well motivated (and ANF is an obvious form to enforce this restriction).

In work of Clark et al., working over permutations of a supplied Walsh spectrum allows various Walsh values to be fixed at convenient values. This allows criteria such as balance, non-linearity and correlation immunity to be "assigned". Walsh spectra are transformed to polar vector form and it is the Boolean-ness that provides the guidance. Essentially, the function space is the set of all vectors induced under inversion by Walsh spectral permutations. Some are Boolean functions (with elements of $+1$ and -1) whilst others are not. In later work, Stanica et al. [30] applied further restrictions to the Walsh spectra by considering only rotation symmetric functions –again with hitherto unattained results. Both the above works demonstrate in different ways the usefulness of appropriate representations and restrictions.

5 Conclusions

In this work, we have shown how heuristic search methods can be useful to identify non-linear approximations to S-boxes. The technique seems to be quite effective and efficient, reaching approximations which can be very useful for mounting a more sophisticate linear attack than by simply using linear masks.

Here we have chosen a family of higher order approximations over the input bits of an S-box. Restricting Γ_y to be a linear function serves good practical purposes. Also, under the assumption of bit independence the expected bias for the overall approximation should be 0 whatever the characteristics of Γ_x. (This is not essential but is certainly convenient.) Allowing Γ_x –along with projection J– to roam free, as it were, over the space of higher order approximations is, we believe, almost forced by modern cryptographic design! The dangers of linearity are well documented and designers seek to design out linear attacks. In these cases, a good non-linear approximation can substantially improve a classical linear attack against the cipher.

An additional advantage of our technique is that the search complexity is bounded by the number of variables desired in the approximation, and not by the actual size of the S-box. According to our experience, the approximations with highest bias usually depends on a large number of variables, which certainly poses a problem for the case of very large S-boxes. However, this method allows a search to be attempted even in these cases, considering that the space explored depends on how many computational resources one can afford.

Acknowledgements

We would like to express our gratitude to the anonymous reviewers for their insights and comments, which have greatly contributed to enhance the quality of the original manuscript.

References

1. Aoki, K.: The Complete Distribution of Linear Probabilities of MARS' s-box. IACR Eprint Archive (2000), `http://eprint.iacr.org/2000/033.pdf`
2. Brown, L., Kwan, M., Pieprzyk, J., Seberry, J.: Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI. In: Seberry, J., Pieprzyk, J.P. (eds.) AUSCRYPT 1990. LNCS, vol. 453, pp. 36–50. Springer, Heidelberg (1990)
3. Burnett, L., Carter, G., Dawson, E., Millan, W.: Efficient Methods for Generating MARS-like S-boxes. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 50–57. Springer, Heidelberg (2001)
4. Burwick, C., Coppersmith, D., D'Avignon, E., Gennaro, R., Halevi, S., Jutla, C., Matyas Jr., S.M., O'Connor, L., Peyravian, M., Safford, D., Zunic, N.: MARS – A Candidate Cipher for AES. In: Proc. 1st AES Conference, NIST (1998)
5. Chaum, D., Evertse, J.-H.: Cryptanalysis of DES with a Reduced Number of Rounds; Sequences of Linear Factors in Block Ciphers. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 192–211. Springer, Heidelberg (1986)
6. Clark, J.A., Jacob, J.L.: Two Stage Optimisation in the Design of Boolean Functions. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 242–254. Springer, Heidelberg (2000)
7. Clark, J.A., et al.: Almost Boolean Functions: The Design of Boolean Functions by Spectral Inversion. In: CEC 2003, IEEE Computer Society Press, Los Alamitos (2004)
8. Evertse, J.-H.: Linear Structures in Blockciphers. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 249–266. Springer, Heidelberg (1988)
9. Feistel, H.: Cryptography and Computer Privacy. Scientific American 228(5), 15–23 (1973)
10. Fuller, J., Millan, W., Dawson, E.: Efficient Algorithms for Analysis of Cryptographic Boolean Functions. In: AWOCA 2002, Frasier Island, Australia (2002)
11. Harpes, C., Kramer, G.G., Massey, J.L.: A Generalization of Linear Cryptanalysis and the Applicability of Matsui's Piling-Up Lemma. In: Guillou, L.C., Quisquater, J.J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 24–38. Springer, Heidelberg (1995)
12. IBM MARS Team. Comments on MARS's Linear Analysis (2000), `http://www.research.ibm.com/security/mars.html`
13. Kaliski, B.S., Robshaw, M.J.B.: Linear Cryptnalaysis using Multiple Approximations. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 26–39. Springer, Heidelberg (1994)
14. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220(4598), 671–680 (1983)
15. Knudsen, L.R., Robshaw, M.J.B.: Non-Linear Approximations in Linear Cryptanalysis. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 224–236. Springer, Heidelberg (1996)
16. Knudsen, L.R., Raddum, H.: Linear Approximations to the MARS S-box. NESSIE Public Report NESSIE/DOC/UIB/001A/WP3 (2000)
17. Matsui, M.: Linear Cryptanalysis Method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
18. Matsui, M.: The First Experimental Cryptanalysis of the Data Encryption Standard. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 1–11. Springer, Heidelberg (1994)
19. Matsui, M.: On Correlation between the Order of S-boxes and the Strength of DES. In: EUROCRYPT 1994. LNCS, vol. 850, pp. 366–375. Springer, Heidelberg (1995)
20. Millan, W., Clark, A., Dawson, E.: Smart Hill-Climbing Finds Better Boolean Functions. In: Han, Y., Quing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 149–158. Springer, Heidelberg (1997)

21. Millan, W., Clark, A., Dawson, E.: Heuristic Design of Cryptographically Strong Balanced Boolean Functions. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 489–499. Springer, Heidelberg (1998)
22. Millan, W.: How to Improve the Non-Linearity of Bijective S-boxes. In: Boyd, C., Dawson, E. (eds.) ACISP 1998. LNCS, vol. 1438, pp. 181–192. Springer, Heidelberg (1998)
23. Millan, W., Burnett, L., Carter, G., Clark, A., Dawson, E.: Evolutionary Heuristics for Finding Cryptographically Strong S-boxes. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 489–499. Springer, Heidelberg (1998)
24. Millan, W., et al.: Evolutionary Generation of Bent Functions for Cryptography. In: CEC 2003, IEEE Computer Society Press, Los Alamitos (2003)
25. Nakahara, J., Preneel, B., Vandewalle, J.: Experimental Non-Linear Cryptanalysis. COSIC Internal Report, 17 pages. Katholieke Universiteit Leuven (2003)
26. National Bureau of Standards (NBS). Data Encryption Standard. U.S. Department of Commerce, FIPS Publication 46 (January 1977)
27. National Institute of Standards and Technology (NIST). Data Encryption Standard. FIPS Publication 46-2. (December 30, 1993)
28. Reeds, J.A., Manferdelli, J.L.: DES Has no Per Round Linear Factors. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 377–389. Springer, Heidelberg (1985)
29. Robshaw, M., Yin, Y.L.: Potential Flaws in the Conjectured Resistance of MARS to Linear Cryptanalysis. In: Manuscript presented at the rump session in the 3rd AES Conference (2000)
30. Stanica, P., Maitra, S., Clark, J.A.: Results on Rotation Symmetric Bent and Correlation Immune Boolean Functions. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 161–177. Springer, Heidelberg (2004)

A Best Approximations Found

$\dim(J) = 8$	
$\Gamma_X =$	C3CB0E857D575014CE88552F31EC89AA02489173571719BB7EB48E96C366CD1F
$J =$	(1, 5, 3, 8, 6, 7, 2, 0)
$\Gamma_Y =$	64393DD1
$\epsilon =$	151/512
$\Gamma_X =$	BDAE5075DB42A75D9279DD3358A4E8907A4C87D61B85D7D137BF7C6DBDF33B71
$J =$	(2, 3, 4, 6, 7, 0, 8, 5)
$\Gamma_Y =$	ECACB346
$\epsilon =$	148/512
$\dim(J) = 7$	
$\Gamma_X =$	AF65A106E589F470E043D55CFEAED634
$J =$	(7, 0, 6, 4, 8, 1, 3)
$\Gamma_Y =$	1387BAF2
$\epsilon =$	118/512
$\Gamma_X =$	1DE5BC329F1B44356E08BEEA44B48F86
$J =$	(2, 4, 5, 6, 1, 3, 0)
$\Gamma_Y =$	3718E4F8
$\epsilon =$	118/512
$\dim(J) = 6$	
$\Gamma_X =$	09778AA1F491AD47
$J =$	(7, 8, 3, 4, 2, 0)
$\Gamma_Y =$	87AEA17C
$\epsilon =$	93/512
$\Gamma_X =$	F4E4ADD42AEAF2B9
$J =$	(4, 7, 2, 3, 5, 1)
$\Gamma_Y =$	CD6D89CC
$\epsilon =$	92/512

B Simulated Annealing

Simulated Annealing [14] is a search heuristic inspired by the cooling processes of molten metals. Basically, it can be seen as a basic hill-climbing coupled with the probabilistic acceptance of non-improving solutions. This mechanism allows a local search that eventually can escape from local optima.

The search starts at some initial state (solution) $S_0 \in \mathbb{S}$, where $\mathbb{S}$ denotes the solution space. The algorithm employs a control parameter $T \in \mathbb{R}^+$ known as the temperature. This starts at some positive value T_0 and is gradually lowered at each iteration, typically by geometric cooling: $T_{i+1} = \alpha T_i,\ \alpha \in (0, 1)$.

At each temperature, a number MIL (Moves in Inner Loop) of neighbour states are attempted. A candidate state C in the neighbourhood $N(S_i)$ of S_i is obtained by applying some move function to S_i. The new state is accepted if its better than S_i (as measured by a fitness function $F : \mathbb{S} \to \mathbb{R}$). To escape from local optima, the technique also accepts candidates which are slightly worse than S_i, meaning that its fitness is no more than $|T \ln U|$ lower, with U a uniform random variable in $(0, 1)$. As T becomes smaller, this term gets closer to 0, so as the temperature is gradually lowered it becomes harder to accept worse moves.

```
$S \leftarrow S_0$
$T \leftarrow T_0$
**repeat until** stopping criterion is met
    **repeat** MIL **times**
        Pick $C \in N(S)$ with uniform probability
        Pick $U \in (0,1)$ with uniform probability
        **if** $F(C) > F(S) + T \ln U$ **then**
            $S \leftarrow C$
    $T \leftarrow \alpha T$
```

Fig. 4. Basic Simulated Annealing for maximization problems

The algorithm terminates when some stopping criterion is met, usually after a fixed number MaxIL of inner loops have been executed, or when some maximum number MUL of consecutive inner loops without improvements have been reached. The basic algorithm is shown in Figure 4.