Efficient Methods for Generating MARS-Like S-Boxes

L. Burnett, G. Carter, E. Dawson, and W. Millan

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

Abstract. One of the five AES finalists, MARS, makes use of a 9x32 s-box with very specific combinatorial, differential and linear correlation properties. The s-box used in the cipher was selected as the best from a large sample of pseudo randomly generated tables, in a process that took IBM about a week to compute. This paper provides a faster and more effective alternative generation method using heuristic techniques to produce 9x32 s-boxes with cryptographic properties that are clearly superior to those of the MARS s-box, and typically take less than two hours to produce on a single PC.

1 Introduction

The Data Encryption Standard [7] has, for the past 25 years, been the US standard for symmetric (shared-key) encryption. In recent years, however, its block and key length have proved to be incapable of providing the levels of security required for applications which utilise shared key encryption. The call for a new standard to replace the Data Encryption Standard for shared-key encryption has been a controversial issue within the cryptographic community for the past two years. The new standard to be known as the Advanced Encryption Standard (AES) [8] has currently been narrowed down to five candidates out of the fifteen initial submissions to the call for AES algorithms in 1997.

The security of a block cipher rests almost entirely on the strength of the components of which it is comprised. These components must not only be secure individually, but must also achieve a much higher level of security when organised together as a cipher system. Substitution boxes (s-boxes) are one of the most important components of a block cipher. They contribute a variety of strengthening properties to the cipher as a whole. Their basic mechanism of allowing bits coming in to an s-box to be replaced with bits going out of an s-box makes them an obvious (and often the only) means of providing nonlinearity to a cipher.

One of the five final candidates for the AES is the MARS cipher [3]. IBM, the designers of the MARS symmetric block cipher, have generated a 9x32 s-box

B. Schneier (Ed.): FSE 2000, LNCS 1978, pp. 300–313, 2001.

which is used in various stages of its computations, both as a 9x32 s-box and as two 8x32 s-boxes. A large effort was undertaken on the part of the designers to generate an s-box which satisfied the very specific properties outlined in the MARS documentation. This required an approximate program execution time of "about a week". However, as will be shown in Section 3, the MARS s-box does not satisfy all the required properties.

In this paper we present an alternative approach to the generation of MARS-like s-boxes using a heuristic technique known as hill climbing. We discuss the cryptographic properties achieved by hill climbing, and in particular give a comparison between these and the property requirements of the MARS s-box. We show that in order for our hill climbing application to satisfy the requirements of the MARS s-box, the program execution time for generation of an s-box was *at most* 3.3 hours. The average generation time for a 9x32 MARS-like s-box using our approach was approximately 2 hours. Apart from speed, hill climbing provides individual output functions that have cryptographic properties superior to those of the MARS s-box output functions.

The remainder of this paper is set out as follows: In Section 2 we outline some important fundamentals in s-box theory. In Section 3 we comment on the cryptographic property requirements of the MARS s-box. Section 4 discusses the techniques used by the designers of MARS to generate the 9x32 s-box used in their computations. We also describe our alternative technique for generating MARS-like s-boxes which satisfy the same requirements imposed by the MARS designers. In Section 5 we discuss some possible variations to our generation technique and relationships between requirements for the MARS s-box and our s-box. Some concluding points are put forward in Section 6, together with some directions for future research in this area.

2 S-Box Fundamentals

An MxN substitution box (s-box) is a mapping from M input bits to N output bits. There are 2^M inputs and 2^N possible outputs for an MxN s-box. A 9x32 s-box, such as the MARS s-box, has 9 input bits and thus $2^9 = 512$ possible inputs. Each input maps to a 32-bit output word.

S-boxes can also be considered as an ordered set of single output boolean functions. The truth table of a boolean function $f(x)$ is a vector containing 2^M elements, each element $\in \{0,1\}$. The polarity truth table of a boolean function, denoted $\hat{f}(x)$, is a simple translation from the truth table where every element 0 in the truth table is replaced by the element 1 in the polarity truth table and every element 1 in the truth table is replaced by the element -1 in the polarity truth table. The relationship can be defined as $\hat{f}(x) = 1 - 2f(x)$.

The hamming distance between two boolean functions $f(x)$ and $g(x)$ is the number of truth table positions in which they are different. The Walsh Hadamard Transform (WHT) of a boolean function, denoted $F(\omega)$, is defined as

$F(\omega) = \sum_x f(x)L_\omega(x)$ where $L_\omega(x)$ is the linear function selected by ω

and gives a measure of the correlation between a boolean function and the set of all linear functions and their complements. Note that a linear function, $L(x) = \sum a_i x_i$ with $a_i \in \{0, 1\}$.

Boolean functions (and therefore s-boxes) are required to exhibit cryptographic properties in order for them to effectively resist certain cryptanalytic attacks. We briefly describe below some of these properties.

A boolean function of M input variables which contains 2^{M-1} ones in its truth table is said to be **balanced**. This property ensures that there is no bias in the truth table. The advantage of using balanced boolean functions is that they cannot be approximated by a constant function. Thus balance is a desirable property to achieve in boolean functions.

The **nonlinearity** of a boolean function $f(x)$ of M variables is given by

$N_f = \frac{1}{2}$ x $(2^M - WH_{max})$

where WH_{max} represents the maximum absolute value of the Walsh Hadamard Transform. The nonlinearity of a boolean function is the minimum Hamming distance to the set of all affine (linear) boolean functions. By this definition a boolean function with high nonlinearity cannot be well approximated by a linear function, thus making the function more resistant to linear cryptanalysis. For this reason, nonlinearity is considered to be one of the most important cryptographic properties of boolean functions.

The autocorrelation function, denoted $\hat{r}_f(s)$, of $\hat{f}(x)$, the polarity truth table of $f(x)$, can be expressed as

$\hat{r}_f(s) = \sum_x \hat{f}(x)\hat{f}(x \oplus s)$.

This cryptographic property provides a measure of the imbalance of all first order derivatives of the boolean function $f(x)$. A boolean function with low autocorrelation makes it more resistant to differential cryptanalysis in that the lower the autocorrelation value, the more difficult it is to approximate the function's first order derivatives.

An M-variable function $f(x)$ is said to be k^{th} order correlation immune, denoted CI(k), if it is statistically independent of the subset $x_{i_1}, x_{i_2}, ..., x_{i_k}$ of input variables where $1 \leq k \leq M$. In terms of security, the output of a correlation immune boolean function, reveals no information about small subsets of input values.

As boolean functions are the building blocks of s-boxes, it is typical to require the same cryptographic properties to be present in s-boxes to improve their strength and ability to resist existing cryptanalytic attacks as well as possible future attacks.

3 MARS Property Requirements

3.1 MARS Differential Requirements

The designers of the MARS cipher, in designing their 9x32 s-box, placed particular emphasis on ensuring that their s-box satisfied a number of property requirements. In this subsection, we discuss these requirements with respect to the combinatorial/differential properties of the s-box and also point out where the s-box does not satisfy one of the stated properties.

Note that they have referred to their 9x32 s-box as S[i], $0 \leq i \leq 511$. This s-box may be divided into two 8x32 s-boxes, S0[j] and S1[j], where $0 \leq j \leq 255$. For ease, we will adopt this notation here also.

Differential Requirements from [3]

1. S does not contain either the word 0x00000000 (all zeros word) or the word 0xffffffff (all ones word).

2. Every pair of distinct entries in each of the two 8x32 s-boxes, S0 and S1, differs in at least three out of four bytes. Equivalently, a pair of words from the same 8x32 s-box may have no more than one byte the same, in the same position.

3. The 9x32 s-box, S, does not contain two entries S[i] and S[j] ($i \neq j$) such that:

i) S[i] = S[j];
ii) S[i] = ¬ S[j]; or
iii) S[i] = -S[j].

In other words, there are no two entries in S which are (i) identical; (ii) are complements of each other; and (iii) sum modulo 2^{32} to give zero.

4. (i) The xor difference of each distinct pair of entries in S is unique and (ii) the subtraction difference of each distinct pair of entries in S is unique.

5. Each distinct pair of entries in S differs by at least four bits.

An examination of the way the s-box entries of MARS are incorporated into the cipher reveals why requirements 1 - 5 are important. An input value selects an s-box entry and this entry is either xored with, added modulo 2^{32} to, or subtracted modulo 2^{32} from an intermediate value. By excluding the all zero subblock, any xor operation involving an s-box entry and an intermediate value changes the intermediate value. In addition, exclusion of the all one subblock ensures that not all bits of an intermediate value are altered. Requirements 2 and 5 ensure that any two distinct entries in S are somewhat different at both the byte and bit level. Requirement 3 ensures that the effect of one s-box entry

cannot be cancelled by another entry from the s-box. Requirement 4 ensures that each possible output "difference" of the s-box is equiprobable, i.e. the Difference Distribution Table is flat.

3.2 MARS Linear Requirements

In this subsection, we discuss the linear requirements imposed by the designers of MARS for their 9x32 s-box.

We note that the linear correlation properties of any MxN s-box can be considered as a (2^M)x$(2^N$-1) matrix where the columns are the Walsh-Hadamard transform vectors of the boolean functions formed by xoring all non-empty subsets of the output functions. Thus the linear requirements can be restated as bounds on the values taken in this linear correlation matrix [2]. Since M is large, even calculating this matrix is very expensive, however we may calculate any individual column we like. The MARS linear requirements are all bounds on particular column subsets of this matrix, which can be calculated easily. It should be noted that the vast majority of the s-box linear correlation columns are not considered in any way by the MARS linear correlation requirements.

In addition, there are strict limitations on the values that can be taken for the correlations between boolean functions, and hence also for the values of bias that can occur. The bias values for an M-input boolean function can only take rational values that are a multiple of 2^{-M}. Thus the choice of bias values $\frac{1}{30}$ and $\frac{1}{22}$ in the property requirements needs explanation.

Linear Requirements from [3]

1. Parity Bias:

The parity bias of S given by $|Pr_x[\text{parity(S[x])} = 0]$ - $\frac{1}{2}|$ is to be at most $\frac{1}{32} = 0.03125$.

This requirement is a bound on the magnitude of the imbalance of the boolean function formed by xoring all output functions. This property thus affects only one column of the linear correlation matrix, that column being the xor sum of all 32 output boolean functions.

2. Single-bit Bias:

The single-bit bias of S given by $|Pr_x[\text{S}[x]_i = 0]$ - $\frac{1}{2}|$ $\forall$ i is to be at most $\frac{1}{30} \approx 0.03333$.

This requirement places a bound on the magnitude of the imbalance for all of the individual output functions. Thus 32 columns of the linear correlation matrix are affected.

3. Two Consecutive Bits Bias:

The two consecutive bits bias of S given by $|Pr_x[\mathrm{S}[x]_i \oplus \mathrm{S}[x]_{i+1} = 0]$ - $\frac{1}{2}|$ $\forall$ i is to be at most $\frac{1}{30} \approx 0.03333$.

This requirement places a bound on the magnitude of the imbalance of boolean functions formed by the xor of two adjacent outputs. There are 31 of these pairs, hence 31 matrix columns are affected.

4. Single-bit Correlation

The single-bit correlation of S given by $|Pr_x[\mathrm{S}[x]_i = x_j]$ - $\frac{1}{2}|$ $\forall$ i,j is to be minimised. The single-bit correlation bias for the MARS s-box is less than $\frac{1}{22} \approx 0.04545$.

This requirement seeks to minimise the correlation that all output functions have with the individual input bits. This requirement affects 32 x 9 = 288 matrix columns.

In all, only a maximum of 352 matrix columns out of 2^{32} - 1 are considered by the MARS linear requirements. With these same requirements, we are able to show that our heuristic processes are able to generate better properties, much quicker.

3.3 Satisfaction of MARS Properties

We shall now discuss the extent to which the MARS s-box was able to achieve the above properties.

MARS S-Box, S comprised of S0 and S1

S satisfies differential conditions 1, 3 and 5. S0 and S1 both satisfy differential condition 2. S does not satisfy differential condition 4. In [3], the authors state that S has 130816 distinct xor-differences and 2 x 130816 distinct subtraction-differences. This is not the case. S has 130813 distinct xor-differences and 2 x 130808 distinct subtraction-differences as evidenced below. In each equation, the xor/subtraction difference of the indexed words on the left is equal to the xor/subtraction difference of the indexed words on the right.

S[27] $\oplus$ S[292] = S[101] $\oplus$ S[360]
S[27] $\oplus$ S[101] = S[292] $\oplus$ S[360]
S[27] $\oplus$ S[360] = S[101] $\oplus$ S[292]
S[13] - S[138] = S[364] - S[297]
S[13] - S[364] = S[138] - S[297]
S[19] - S[168] = S[509] - S[335]
S[19] - S[509] = S[168] - S[335]
S[49] - S[142] = S[97] - S[392]
S[49] - S[97] = S[142] - S[392]

S[333] - S[131] = S[211] - S[348]
S[333] - S[211] = S[131] - S[348]

The parity bias of S is 2^{-7}, as stated in the MARS paper, which is less than the threshold value of $\frac{1}{32}$. The single-bit bias of S is at most ≈ 0.033203 which is slightly less than the limit of $\frac{1}{30} \approx 0.033333$. The two consecutive bits bias of S is at most ≈ 0.032290 which is less than the bound of ≈ 0.033333. The maximum single-bit correlation bias of S is about $0.044922 < 0.0454545$, as stated in the MARS paper. Thus all linear conditions imposed by the designers of the MARS s-box are satisfied by S.

4 S-Box Generation Techniques

4.1 Summary of MARS S-Box Generation Techniques

As mentioned earlier, the MARS s-box is a 9x32 s-box containing 512 32-bit entries. The approach taken by the designers of the s-box was to generate the 9x32 s-box by using the well known SHA-1 (Secure Hash Algorithm-1) [9]. SHA-1 produces a 160-bit digest comprised of the concatenation of five 32-bit words. The input used for SHA-1 is the value 5i|c1|c2|c3 where i = 0..102, cj (j ∈ 1,2) are the fixed constants

c1 = 0xb7e15162
c2 = 0x243f6a88

and c3 is allowed to vary until the first eight property requirements are satisfied. The value for c3 which minimises requirement nine is then the one chosen. Therefore, each entry of the 9x32 s-box, S is computed as follows:

S[5i+k] = SHA-1(5i|c1|c2|c3)$_k$

denoting the kth word of the output of SHA-1 (k = 0..4, i = 0..102).

The designers started the computational process with c3 = 0, increasing its value until the resulting s-box was found. Each value of c3 resulted in a 9x32 s-box which was divided into two 8x32 s-boxes. For each value of c3, the xor sum of distinct pairs in S0 and S1 was checked to see if it contained more than one zero byte. If this was the case, then S[i] was replaced by $3 \cdot$ S[i] for one of the words S[i] in the pair. The new s-box was again tested for the 5 differential requirements and first 3 linear requirements. If this test was passed then the single-bit correlation was calculated. The final fixed constant value of c3 was 0x02917d59. This value was found to best minimize the single-bit correlation.

As stated in [3], the program for generating S ran for about a week, with the value of c3 increasing to 0x02917459 = 43 086 937$_{10} < 2^{26}$. The MARS s-box can be found in [3].

4.2 Summary of our Techniques for Generating MARS-like S-Boxes

Our approach to generating MARS-like s-boxes is a flexible one which allows for much variation in parameters and heuristic methods used. The particular technique we chose was a heuristic method known as hill climbing [6].

4.2.1 Hill Climbing

Making small changes to the truth table of a boolean function produces one of three effects on the WHT of the function - the WHT values can decrease, remain unchanged or increase. In terms of properties such as nonlinearity, this means that the nonlinearity measure of the new boolean function resulting from the change can either become smaller, remain the same or become larger. Hill climbing takes advantage of this effect to optimise cryptographic properties of boolean functions (and thus s-boxes) by retaining a change which has brought about an improvement in a property value, such as nonlinearity. Such an improvement is incremental and consequently explains the analogy with climbing a hill.

Essentially, hill climbing involves the following steps:

1. Measure the property of concern for the original function.
2. Select a pair of elements to complement ensuring that the pair chosen consists of a zero and a one. (This ensures balance is maintained).
3. Measure the property of concern for the new function.
4. If the measure of the property in 3 is 'better' than the measure of the property of the original function, then accept this new function as the original function. If the measure of the property in 3 is worse, then retain the original function.
5. Repeat steps 2, 3 and 4 until a predetermined stopping criteria has been reached.

4.2.2 General Procedure

The technique we used to create our s-boxes began with the generation of random single-output balanced boolean functions. Each boolean function was hill climbed to reach a minimum nonlinearity value, a parameter allowed to vary for optimum results. The goal of this approach was to generate a set of 32 balanced boolean functions which not only each achieved the minimum nonlinearity value set by the user, but was also constrained by a maximum imbalance limit between pairs of boolean functions and was further constrained by a maximum deviation limit from CI(1).

A set of 32 boolean functions achieving these limits comprise a 8x32 s-box containing 256 words. Pairs of s-boxes of this size were combined to form a 9x32 s-box. It seemed less complicated to generate 9x32 s-boxes in this way due to the necessity of satisfying certain requirements placed on the 8x32 s-boxes individually. The s-box was then checked for the differential and linear

requirements placed on the MARS s-box. In order to satisfy differential condition 2, it was necessary to modify a small number of bytes in each of the 8x32 s-boxes, typically in less than half a dozen entries, and re-checking that condition, particularly for previous pairs of entries. Similarly, the satisfaction of differential requirement 4 involved replacing a small number of entries in the 9x32 s-box. Subsequently, the new s-box was tested for all 9 conditions again. We ensured that the introduction of any replacement entries in the s-box did not destroy the balance property achieved by the initial functions.

4.3 Experimental Results

We stated in Section 3 the differential and linear requirements, together with threshold values set by the designers of MARS for their s-box. We shall now discuss the extent to which our s-box was able to achieve these properties.

Our S-Box, SB comprised of Sb1 and Sb2

Note that SB[i] where i = 0..511 is a 9x32 s-box, and Sb1[j], Sb2[j] where j = 0..255 are both 8x32 s-boxes. SB can be found in **APPENDIX A** and also at http://www.isrc.qut.edu.au/papers/2000/AppendixA.txt.

SB satisfies differential conditions 1, 3, 4 and 5. Sb1 and Sb2 both satisfy differential condition 2 of the MARS s-box requirements.

The parity bias of SB is 0.019531 which is less than the threshold value of $\frac{1}{32}$ = 0.03125. The single-bit bias of SB is zero. The absence of any single-bit bias in SB is due to the balance in each of the 32 boolean functions which comprise the s-box. The two consecutive bits bias of SB is at most ≈ 0.024462 which is less than the bound of ≈ 0.033333. The maximum single-bit correlation bias of SB is $0.03125 < 0.0454545$. Consequently, all linear conditions imposed by the designers of the MARS s-box are satisfied by our example 9x32 s-box, SB.

The achievement of these results depended largely on the three parameters used in our s-box generation program. For our experiments, we typically generated sets of 32 boolean functions with a minimum nonlinearity of 110, although we experimented with parameters above and below this value. A parameter value for minimum nonlinearity at around 108 produced 8x32 s-boxes in less than 10 minutes, while minimum nonlinearities of 112 for an 8x32 s-box took about 3 to 4 hours to generate. Our second parameter was a limit on the maximum imbalance between distinct pairs of boolean functions in the set. A typical parameter value for this limit used in our computations was 10. It was desirable to have a low imbalance between pairs of boolean functions which consequently had the effect of reducing the two consecutive bits bias condition imposed by the MARS s-box designers. We also placed a large degree of importance on our third parameter, the maximum deviation from CI(1). By minimizing this parameter value over all boolean functions, we were easily able to produce a single-bit correlation value below the given bound. Typically, we used parameter values such as 16 or 24 to be the maximum allowable deviation from CI(1) for the 32 boolean functions comprising the 8x32 s-boxes.

A combination of hill climbing and appropriate setting of the parameters discussed above allowed us to produce good 8x32 s-boxes, pairs of which gave us 9x32 s-boxes. Most of the s-boxes generated by our technique were very close to satisfying the MARS s-box requirements. In fact, for those s-boxes which we successfully generated, bias values, when exceeded, were so by only extremely small margins. The remaining s-boxes which we generated were easily able to satisfy the same conditions that the MARS s-box satisfies.

Based on a heuristic technique approach, we were able to generate a number of MARS-like s-boxes with little effort. In addition, the program execution time, depending on the parameters chosen, varied from approximately 16 minutes to around 3 hours and 20 minutes on a single Pentium II 300 MHz PC. This time frame is a huge improvement on the program running time for the MARS s-box of about a week.

5 Property Relationships and Technique Variation

5.1 Property Relationships

An s-box comprised of balanced boolean functions clearly possesses no single-bit bias since the number of ones and zeros in the truth table of balanced boolean functions is the same. Our s-box generation procedure began with the generation of a set of 32 balanced boolean functions. In order to satisfy differential requirements 2 and 4 it was necessary to replace a small number of bytes and entries respectively. However, at all times throughout our computations we retained balance in the boolean functions. None of the boolean functions comprising the MARS s-box are balanced. However, their deviation from balance is not large enough to violate the single-bit bias requirement.

Nonlinearity is a very important cryptographic property of single output boolean functions and s-boxes. Higher nonlinearity indicates a reduction in the magnitude of statistical correlations between sets of input bits and sets of output bits. The nonlinearity of an s-box is measured by the magnitude of the largest Walsh Hadamard Transform (WHT) value in the linear correlation matrix. The linear requirements in [3] are concerned solely with balance properties and no requirements on nonlinearity values are given. The nonlinearity of the individual boolean functions in Sb1 and Sb2 ranges from 108 to 112 inclusive, with an average nonlinearity of 110. The boolean functions comprising the MARS s-boxes, S0 and S1, have nonlinearity values ranging from 92 to 109, the most frequent nonlinearity value being 102.

Although in our s-box generation procedure we have not directly sought to optimise the autocorrelation property, the boolean functions comprising our s-boxes have, in general, displayed low autocorrelation values. A low autocorrelation distribution for an s-box serves to improve its differential properties, in particular, by flattening the Difference Distribution Table. The range of autocorrelation values for the individual boolean functions in our 8x32 s-boxes, Sb1 and Sb2, was between 48 and 88, averaging around 56. We note that the boolean

functions comprising the MARS s-boxes, S0 and S1, displayed autocorrelation values of between 52 and 88, averaging 64.

One of the three important parameters set by our code was a limit for the maximum imbalance of the xor sum of distinct pairs of boolean functions. The purpose of this restriction on boolean function possibilities was to reduce the imbalance between pairs. A low level of imbalance between pairs of boolean functions includes the effect of reducing the two consecutive bits bias i.e. the bias between adjacent output boolean functions.

Our requirement for setting a maximum deviation from CI(1) for the individual boolean functions is identical to the single-bit correlation requirement placed on the MARS s-box. Minimising this measure reduces the magnitude of correlations which exist between individual input and individual output bits of the s-box.

5.2 Possible Variations on our Techniques

A great number of variations to our technique for generating MARS-like s-boxes may be adopted as alternative approaches to this task. An obvious generation method would be to apply another useful heuristic technique called the genetic algorithm to randomly generated boolean functions in order to "build" a cryptographically strong 9x32 s-box. Genetic algorithm applications have been very successful in improving cryptographic properties of boolean functions and s-boxes. Indeed, in [5] it was found that a combined genetic algorithm with hill climbing proved to be even more successful in generating boolean functions with good cryptographic properties such as nonlinearity and autocorrelation.

Additional parameters may be included in the code for the generation of a stronger s-box, for example, criteria for strict avalanche and propagation. Varying the parameters used in the generation process allows for a different strength emphasis in the resulting s-box, although the reader should note the existence of conflicting properties which affect each other in a negative way.

Further, it should be noted that only a small subset of the linear correlation matrix is utilised by the linear requirements imposed by the designers of the MARS s-box. However, as a consequence of our parameter choices a larger subset of the linear correlation matrix is utilised in the generation of our s-boxes, thus making greater use of the information contained in this matrix. We believe that an even stronger s-box can be generated if more information from this matrix is incorporated into s-box design. However, it should be noted that to generate the complete linear correlation matrix and analyse it in its entirety is not practical due to the computational effort required for this task.

6 Conclusions and Future Research

The designers of the MARS s-box have successfully generated a 9x32 s-box which satisfies all but one of the requirements placed on it relating to differential and linear properties. Their s-box failed to satisfy differential condition 4, despite

claims that it did in fact do so. A long search running through values for c3 caused the program to take about a week to produce the final MARS s-box. In this paper we have presented an alternative approach to the generation of MARS-like s-boxes providing satisfaction of all of the requirements which were placed on the MARS s-box. Further, we have shown that by using a combination of random boolean functions, heuristic techniques and appropriate parameters we have gained additional properties such as higher nonlinearity and balance. This approach requires far less effort and compares very favourably to the MARS approach particularly in terms of computation time and ease of generating not only one but a number of MARS-like s-boxes.

Much work is needed to be done in this area in order to conduct an indepth investigation into the ways in which s-boxes with good cryptographic properties may be generated. The desirable properties are not only limited to those of the differential and linear type, even though their importance, stemming from targets of powerful cryptanalytic attacks, is by no means trivial. In the previous section we have endeavoured to outline a few of the variations on our techniques which could be investigated. Work directed towards other optimisation techniques for improving the cryptographic properties of s-boxes may be another worthwhile path for future research in this area. We believe that the strongest s-box will be one which achieves the correct balance between cryptographic properties. It is an open problem to find this balance.

References

1. E. BIHAM and A. SHAMIR Differential Cryptanalysis of DES-like Cryptosystems *Journal of Cryptology*, 4:3-72, 1991.
2. J. DAEMEN, R. GOVAETS and J. VANDEWALLE Correlation Matrices *Fast Software Encryption*, LNCS vol. 1008, pages 275-285, Springer-Verlag, 1994.
3. IBM Corporation MARS - a candidate cipher for AES *http://www.research.ibm.com/security/mars.html*.
4. M. MATSUI Linear Cryptanalysis Method of DES Cipher *Advances in Cryptology - Eurocrypt'93*, LNCS vol. 765, pages 386-397, Springer-Verlag, 1993.
5. W. MILLAN, A. CLARK and E. DAWSON An Effective Genetic Algorithm for Finding Highly Nonlinear Boolean Functions *International Conference on Information and Communications Security, ICICS '97 Lecture Notes in Computer Science* Vol. 1334, pages 149-158, Springer-Verlag, 1997.
6. W. MILLAN, A. CLARK and E. DAWSON Smart Hill Climbing Finds Better Boolean Functions *Workshop on Selected Areas of Cryptology, SAC '97, Proceedings*, pages 50-63, 1997.
7. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) Data Encryption Standard *U.S. Department of Commerce FIPS* Publication 46, January 1977.
8. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) Request for Candidate Algorithm Nominations for the Advanced Encryption Standard (AES) *Federal Register* Vol. 62 No. 177, pages 48051-48058.

9. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) Secure Hash Standard *NIST FIPS PUB 180*, U.S. Department of Commerce, May, 1993.

APPENDIX A

An example S-box generated by our techniques using heuristic methods is SB[i] below where i = 0..511:

0x657ce571	0xb2c0a31b	0xeaaacac0	0xd4d49175	0x4794396c	0xada63322
0xb6476df8	0x5d8b1bdb	0x3216bd0c	0x87810f0e	0x8928aab6	0x309926d6
0x86ed7cda	0x7ce28025	0xab91f5e5	0xa0559c17	0x03b7fcc1	0xc635a7c2
0xb12e7967	0xf3c464ce	0x1c8815d1	0x12fa97fb	0x6937c3b8	0xd8f7406d
0x581a310f	0xf60add94	0x3e297a67	0x61ecf4be	0x4abcb39d	0x3fbf5af2
0xb01c48c9	0x01193559	0xae0d8259	0xd1229472	0x2a1c3b84	0xd3b54059
0xd1557eb8	0x6d1f101c	0xee7fd7ab	0x3ac220a1	0x03e23430	0xd6746be1
0x5e026256	0x57f98f80	0x8b09cf06	0xe503ea7f	0xa3268b2a	0x192bb9e3
0xc28a5f35	0x54c8ceef	0x0ee5da73	0x4d3154a7	0x8eda0ab3	0xe5e18c07
0x9e923d96	0xf94dc633	0xb02e60bc	0x1b6acf89	0xb8c718a2	0xad77b720
0x2444c1d0	0x9d64bd69	0x6c7ea648	0xc6b3f1be	0x85fbf907	0xb62ab1a0
0x105349ff	0x0c0d7808	0xb9af64fd	0x81f3c534	0x1a450da2	0xf5d20e38
0xd8ea00da	0xd149c90a	0x2b69526e	0x9d7d7598	0xefe96d87	0x7f55539a
0x819c2b62	0x7f85c4ff	0x6c8d1bb8	0xcf7b529d	0x664294e4	0x7eb2d2cc
0x6fd7ade7	0x4ee6b926	0xb858f38d	0xc2c47b42	0xbeb2006e	0x75003971
0x1eb2eb50	0x02eff63d	0x05dbe8ce	0x4d0ccac2	0x502fc81f	0x25724c59
0x9852165f	0xa9bd3bb2	0x40308156	0x319ebb09	0x3bb1370f	0x18718f78
0x751ed38a	0xe74acd36	0x59745744	0xda8f3b85	0xf4771cfe	0x6510184d
0xc36d332b	0xbfb8d681	0xe95e9ec7	0xc0332dec	0xcb24e5f4	0x6a746cbb
0xe9a5b509	0x0fbc5c93	0x8b138d45	0x8f6a906e	0xde78fe6b	0x131faa01
0xe79f8558	0x64b15239	0x255e0943	0x7be2d50a	0x6d28a6bc	0xba53449d
0x8cc7e39a	0xd29d82f2	0xd940cab8	0xe0d39beb	0xb079da15	0xdc1d313e
0xcb032e98	0x9e3ff5f0	0x77da39db	0x06cc4b3c	0x6d7323c6	0xc880d552
0x63fd8825	0x98e0d78d	0x6861c1cb	0x710fd4e5	0x79b69e4e	0x00061be5
0x623125bb	0xa54b082c	0xcdc97ce2	0x99f71a6f	0xd1443f73	0xb406ff77
0x04a2f4b4	0x67698cd3	0xaa3d5731	0x59c12151	0x5a9f8068	0xe29e555c
0xefacd992	0x418f3f8f	0xb3233fd9	0xe8c97421	0xe673f889	0x2fd7f4d8
0x5e838793	0x654e53b7	0x20fad86e	0x0729f2ce	0xf788004f	0xbcce24e0
0x1f27ab52	0x49ff2416	0xe6afc9b1	0x09995df5	0x834c7268	0x17daa0cf
0xacc21c23	0xb4f41552	0xf018c993	0xe247cf88	0x11caef8d	0xcab5a62f
0x41bf4a29	0x68147ece	0x16396c17	0x707d2204	0x74b40fac	0xde046da6
0xf2e39b32	0xafa3025d	0x18d2f854	0x1cb5d9ec	0x9fdb4066	0xd755650c
0xe178476e	0x81b6dadb	0x871587fe	0x0e4bfb09	0x7aad28c5	0xf32a077b
0xd3ee8184	0x7db97e78	0x77e03897	0x02d05ec4	0xe4daa729	0x94cedc15
0xc6a41eab	0x1499c20a	0x3f20e0d8	0xf22df536	0x2b2e1c53	0x104dac0e
0xc23faec3	0xacca64e3	0xaaf70012	0x3a498f24	0x21353c82	0x19a00a08
0x8d1016ed	0xa61b6b33	0x3743e626	0x5050a261	0x5dcb8660	0x7338f3a9
0x4e070f37	0xe9b2e637	0x779110a2	0x12792697	0x14457cbb	0x3884e0af
0x3e5a7cb8	0xf0b1a844	0xa4a05227	0x62637655	0x07e4e409	0xad8d8f5c
0x1de7a9fd	0xce4c62d9	0x3f08c7d5	0x5b56524f	0x98a46531	0x228b9fde
0xdc7e2e02	0x0ece49d2	0x853351ed	0xc1687310	0x3b92f374	0xf2cfa8dc

0x8eb66314	0x27ccb3f6	0xf7c9237e	0x4850359c	0x1954b0e1	0x1bb20c31
0x9525bed1	0x046e076b	0xdc54db4c	0x3bfca3b5		
0x558773ab	0x28b2ad90	0xf6973f75	0xaca086bd	0xd7cdc737	0x56815e7d
0x1f0dbc30	0x78ace509	0xa6a0a02c	0x7fd04834	0xc9c4a588	0xee8681ee
0x3b7bb9c7	0x23ed1bd7	0xe26ef6ce	0x8e1f20db	0x1becd6e8	0x972421a6
0x37015bdf	0xd2fa8b20	0x98e1fa8c	0xf33a1d5e	0x2e40af46	0xad43454d
0xa12419d4	0x764b7009	0xf117276e	0x590e0446	0x9c9d0f79	0x464b03be
0x36ee34c1	0xcda77e71	0x318500d2	0x1e5e1033	0x902f9947	0xa98bc045
0xbee34bc2	0xee558b95	0xa033121d	0x299e0222	0xab2b7680	0xda013864
0x8456e39c	0x88f6d2c2	0x20387d0b	0xd59e867b	0xbdc85129	0x29f8a23d
0x05238897	0xa671d73a	0xe5a2749a	0x8498b1b9	0x5284c9a0	0x05a35fb5
0xfd641dc0	0x01e04f29	0x607c222e	0xe37daa81	0x3b504c33	0xd4283597
0xe4bf5c64	0x171271f7	0xb72de4be	0x85d072d7	0x95648343	0xc3fed911
0xe19d9084	0xfd46ddc1	0x5a28301f	0xc5106adc	0x7fdcafaa	0x97864b57
0x26441e1b	0x08fc5943	0x6bb68bf7	0xe0d10d7f	0x2ee6b878	0x62dc5145
0xbdbd3c1e	0x4fe9d606	0x44404830	0xfb8a3222	0xd3d97c07	0x6afd4cad
0x18772ce4	0x6b8a39a0	0x1853f57c	0xcf3894b1	0xae377739	0x1973945c
0x2c3fe2eb	0xd8b1f202	0x65b9cc80	0x11741355	0x8390161c	0x5fc3bf0a
0x3b98ec59	0xb2170ae3	0x04ff1c07	0x483272cc	0xd8e99670	0xec3e63c2
0xd5d8b58b	0x3ac2ec07	0x076be86e	0xa304eec4	0xea534d84	0x7a7a57e6
0xc106953c	0x5bb2ef63	0x06d3d71c	0x2a9fcad7	0x7ac837c9	0x620eeb1a
0xf4692142	0x1a906cab	0x71b25a75	0xd54dc1fe	0xe3791c6b	0x1779abcd
0x49c276fa	0x7735cfef	0xdc8fcaeb	0xb7e1bdc1	0x1e0a0ed4	0x674f3390
0xa506bc60	0x5409c03b	0xc08acd89	0xa01e5ee0	0x3d6b4a50	0x7d9d2477
0x32fabebe	0xba61f9d0	0x32d7f5c6	0x4d6d06f0	0x879ff997	0xa974cf79
0xd9727823	0x46918f88	0x0a19c4ff	0xfeb10189	0x09b4a7da	0x74f24b8c
0x1abd6bd9	0x2fee52ad	0x937b044c	0xfd8e3ccf	0x782d8e2b	0xc9dcd807
0x1ace6736	0xce30c05c	0xc0ab6563	0x98dd5602	0xb31a52a8	0xc3732b2e
0x88633833	0x4f0c5b26	0x30634892	0x6db73b5d	0x4099469f	0x79a5f0b6
0x9e7ecd2a	0x4177a990	0x71929643	0xfc9879e4	0x6f7c6158	0x31bc8ddc
0xda5a8574	0xc64df673	0xfeb43a6d	0x87caf074	0x1c8376ca	0x1a28b32f
0xd6d4e0b3	0x45682e84	0x21d2ce9d	0x72a7fc5e	0x58e87ae0	0x5404a615
0x86244304	0xeaeac14f	0x6b40d113	0xed15e3ab	0xc29f99f9	0x2c75e7ee
0xf364db5f	0x14e1058b	0xee592f43	0xa5e39cb3	0x884bfa85	0x120c3c1f
0x756b17b5	0x375c388b	0x8c1e7fea	0x0113d4e1	0x4b97581b	0xb243ef45
0xd7668abf	0xa32565b2	0x25bd9341	0xc1817258	0xde4136a8	0x5b5994d2
0xd55aaf72	0x2a1e200d	0xb320a251	0xb0c9d7bc	0xb0d9a994	0xd4fae0b9
0xc808f572	0x2cd72112	0xfd3dcd52	0x308db7b4	0x67989372	0xe847aca3
0x2f064d6f	0x34d14d7c	0x74b5a608	0xb7abf1d3	0x6ae58795	0xcc4fb967
0x44fa099c	0xbb56d026	0x8c3d93f7	0xde33075c	0x0221622a	0x592255ea
0xa1cefd2e	0xababb33d	0xc64f1a49	0x7e301a30	0x398361fb	0xb44f067c
0x0bdaa3a6	0x1d09ecb9	0x146ea2e5	0xd7b7508a	0x48656859	0x67ec3ff0
0x02ef76a8	0x8d97a9ee	0x0edc95e5	0x88be0eb7	0x11e59acf	0x4674ffed
0xb3966531	0x8d7798b8	0x6d021a5e	0xdb86e411	0x78618b37	0xafe9287c
0x9bd7274c	0xdd240d88	0x4412d92a	0x932082c9		