IOSR Journal of Engineering
Apr. 2012, Vol. 2(4) pp: 908-914

An Analytical Approach to S-Box Generation

K. J. Jegadish Kumar[1], K. Hariprakash[2], A.Karunakaran[3]
[1](Department of ECE, SSNCE, India)
[2](Department of ECE, SSNCE, India)
[3](Department of ECE, SSNCE, India)

ABSTRACT
This paper presents the construction of asymmetric and symmetric substitution boxes using an analytical approach. The objective was to generate S-Boxes that meet the Strict Avalanche Criteria (SAC), are non-linear, and have a high degree of resistance to differential cryptanalysis. It was found that it is possible to produce S-Boxes which exhibit robustness up to 0.96, and several were found to give above 70% compliance with the SAC.

***Keywords*: Cryptography, S-Box, Analytical design, Strict avalanche criteria, Robustness**

I. INTRODUCTION
The concept of concealing the meaning of a message from all but the intended recipients is the job of a crypto system. The development of digital computers allowed the complexity of encryption algorithms to increase and to be used in a wide range of applications, including civilian use. Since it is easier to handle data as blocks most of today's popular encryption algorithms are all block ciphers i.e. they work on blocks of data e.g AES , Blowfish. These algorithms are made up of many primitive transformations among which one of the most vital transformations is the 'substitution'. In this step a given value is substituted with another value by use of a lookup table. This imparts very high non linearity, bit dependency etc and makes the encryption algorithm immunity to attacks such as linear and differential cryptanalysis. In this paper we discuss about generating these lookup tables alias S-Boxes using an analytical method where we make use of condition hierarchies and various filling algorithms. The discussion starts with a brief on the different forms of cryptanalysis and move on to the properties of S-Box which enhances its strength. After these basic briefings the paper starts discussion about the generation and analysis of S-Boxes**.**

II. CRYPTANALYSIS
Cryptanalysis refers to the process of attempting to recover the plaintext or the key that corresponds to a particular ciphertext by a party who is not the intended recipient. The most obvious form of cryptanalysis is the Brute Force Attack, also known as an exhaustive key search. This attack is carried out with a plaintext ciphertext pair with keys being tried until the resulting plaintext is equal to the known plaintext. However doing an exhaustive search is not an easy task primarily because the key spaces are very large. It might even take several years to figure out the key through such a raw search. Thus before doing a brute force attack pre analysis such as differential or linear cryptanalysis is done to reduce the probable key space and thus reduce the time taken to a practicable value. A brief description of the two cryptanalysis techniques is given below .

2.1 Differential cryptanalysis
Differential cryptanalysis is a type of attack that involves analysing the changes that occur in the ciphertext when different changes are made to the plaintext. The ciphertext changes are then interpreted with reference to certain exploitable characteristics of the round function. Usually the round function characteristics that are exploited relate to the S-Boxes. Typically a very large number of plaintexts/ciphertext pairs are required for this attack. The objective is to reveal either the whole key or enough of the key to make a bruteforce attack. [1]

2.2 Linear Cryptanalysis
This cryptanalysis attempts to make linear approximations of the operations performed by an algorithm. Probabilities are assigned to each approximation. If enough information is gathered it may be possible to deduce some key bits to within a certain degree of accuracy. A linear cryptanalysis of DES is given in [2].

III. S-BOX ANALYSIS PARAMETERS
Linear and Differential cryptanalysis are very powerful methods for attacking encryption algorithms. Both take advantage of weaknesses found in the round function of an algorithm, usually in the S-Box characteristics. It is desirable that there be techniques for generating S-Boxes that have characteristics which endow them with a degree of resistance to differential and linear cryptanalysis. Two of the important characteristics which decide the strength of an S-Box are Robustness(R) and Strict Avalanche Criteria (SAC), both of which are derived from the Differential Distribution Table (DDT). The construction of the DDT and the procedure to derive the SAC and R is discussed below.

3.1 Differential Distribution Table (DDT)
The Differential Distribution Table (DDT) is a table that indicates how the output of an S-Box varies as the input is varied. The number of rows in the S-Box is equal to

the number of elements in the S-Box (2^n where n is the number of input bits), and the number of columns is equal to the number of distinct output values in the S-Box (2^m, where m is the number of output bits). Each row corresponds to a change in the input value, i.e., the XOR of the input with some other value of equal bit length. Row 1 corresponds to the trivial case of XOR with zero, while the final row corresponds to XOR with n 1's. The columns correspond to the XOR of the original and changed output. Each element in the table indicates how many times a given XOR at the input to the S-Box results in a particular change at the output. The sum of each row is equal to 2^n and the sum of each column $2^{2n}/\ 2^m$. The first row of the table is all zeros except for the first element, which is equal to the number of elements in the S-Box. Obviously when any input to the S-Box is XORed with zero (not changed), the output will not change. The DDT is useful as it allows for the assessment of how well an S-Box conforms to the criteria under which it was generated. The compliance of an S-Box with the Strict Avalanche Criteria (SAC), can be checked by looking at the rows of the table that correspond to an input change of one bit, and the columns that correspond to an output change of half the output bits. In an ideal situation, the sum of all such columns in a one-bit change row would be equal to 2^n. Dividing this sum by 2^n (and multiplying by100%) gives the percentage compliance with the strict avalanche criteria for that bit change. Averaging this value over all one-bit changes gives an indication of the overall compliance of an S-Box with this criterion. The DDT can also be used to check similar properties such as how often a one-bit input change results in only a one-bit output change.

3.2 Robustness (R)

Robustness is a measure of the resistance of an S-Box to differential cryptanalysis. Robustness is based on two features of the DDT. The first is the number of nonzero elements (N), in the first column of the DDT (excluding the first element). These denote instances when a change in the input results in no change in the output. Such occurrences are a weakness as they reduce the complexity of an algorithm and play an important part in differential cryptanalysis. The other feature is the largest value found in the DDT (L) other than the (1, 1) element.

The Robustness is given by

$$R = (1 - N/2^n)(1 - L/2^n)$$

where n is the number of input bits. The higher the value of R the better is the S-Box's resistance to differential cryptanalysis.

IV. DESIRABLE S-BOX CHARACTERISTICS

4.1 High level of compliance with the SAC

It is desirable that the degree to which output bits are dependent on input bits should increase as rapidly as possible through an encryption algorithm. This is primarily an action of the S-Boxes. SAC requires that a change in one input bit results in half the output bits of the S-Box changing. The ultimate goal is that every output bit of an encryption algorithm be dependent on every input bit. This means that changing one of the input bits should result in a new output that is unrelated to the previous output. The requirement is that if 1 input bit changes at least half the output bits must change.

4.2 Non Linearity

It should not be possible to express the operation of the S-Box as a linear function of the inputs. This would allow the encryption algorithm to be broken through a process of solving a set of equations for a set of unknowns. Typically as S-Boxes become larger the probability of them containing any linearity decreases rapidly.

4.3 Resistance to Differential Cryptanalysis Cryptosystems

Differential Cryptanalysis[3] is a method of breaking encryption algorithms that are based on Feistel networks. It is a statistical attack that uses the characteristics of an S-Box
given by its DDT to determine either all the key bits or enough of them to reduce the complexity of a brute force attack to a manageable level. The two main features of an s- box that differential cryptanalysis exploits are:
–large value entries in the DDT,
–entries in the first column of the DDT, particularly those that have large values.
Hence the DDT of an S-Box should have a low maximum value (excluding the (1, 1) entry), which is not significantly greater than the other entries, and a low number of entries in the first column but not so few that they have large values. These last two requirements are conflicting. A small number of entries presents a restricted number of opportunities for differential cryptanalysis to be performed, while small value elements mean that when the cryptanalysis is performed it will be more difficult to get a conclusive result. Having seen the various criteria than makes an S-Box strong, we now proceed to device algorithms to generate such quality S-Boxes using fundamental filling methods.

V. FILLING CONDITIONS

Before stepping into generating S-Boxes we have to define certain conditions which when imposed during generation will result in S-Boxes of desired quality. Logically conditions can be defined as the number of output bit change that is allowed for a particular number change in the input bits. Since it is very difficult to fill an S-Box in such a way that all the elements satisfy a particular desirable condition there comes a need to declare a hierarchy of conditions in decreasing strictness. Filling of each position in the S-Box starts with the search for numbers that satisfy the condition level 1 of the hierarchy. If there is no value

existing that satisfies condition level 1, it moves on to find a numbers that satisfy the next condition level, else it randomly chooses a number from the set of satisfying numbers and fills up that position. Many conditions affect the characteristics of an S-Box, but generally it is the one and two bit input changes that affects the Robustness and SAC to an appreciable level. We narrowed upon three sets of condition hierarchy among which two of them gave good results during analysis. These two hierarchies are tabulated in Table 1 and Table 2. Let α be a function which takes *x* as the input and gives *y* as output. (i.e). $\alpha(x) = y$

Where

x – Number of input bits changed

y – Corresponding number of change in output bits

The Condition Hierarchy I (CH1) can be represented as shown in Table 1. Here level 1 denotes the most desirable condition and level 6 denotes the least desirable condition.

Table 1 : Condition Hierarchy I

Hierarchy Level	Condition
1	$\alpha(1) = 4$ and $3 \leq \alpha(2) \leq 5$
2	$\alpha(1) = 4$ and $2 \leq \alpha(2) \leq 3$
3	$\alpha(1) = 4$
4	$3 \leq \alpha(1) \leq 5$
5	$2 \leq \alpha(1) \leq 6$
6	$1 \leq \alpha(1) \leq 7$

These conditions were formulated primarily to increase the SAC and Robustness of the S-Box. Importantly the first two conditions which has the α(2) conditions were introduces to spread the values in DDT and thus help in decreasing the value of L which thereby increase R.

The Condition Hierarchy II (CH2) is shown below in Table 2.

Table 2 Condition Hierarchy II

Hierarchy	Condition
1	$\alpha(1) = 4$
2	$3 \leq \alpha(1) \leq 5$
3	$2 \leq \alpha(1) \leq 6$
4	$1 \leq \alpha(1) \leq 7$

This condition hierarchy is more lenient as compared to CH1. We have made such a modification to give more importance to the increase in SAC with a small compromise in R. Further it also reduces the time taken to generate these S-Boxes by a good margin. Its effect can be observed in our analysis which is to follow.

VI. GENERATION AND ANALYSIS OF ASYMMETRIC S-BOXES

There are many kinds of S-Boxes. Basically they are defined as *n* x *m* where *n* is the number of input bits and *m* is the number of output bits. For example the Data Encryption Standard (DES) used a 6x4 S-Boxes and the presently used standard the Advanced Encryption Standard (AES) uses an 8x8 S-Boxes. These S-Boxes are not symmetric i.e not invertible $S(S(x)) \neq x$. A lot of algorithms have been developed to create such non-symmetric S-Boxes, for example the Rijndael's method which uses the Affine Transforms for generating the S-Box and its inverse. We for generation and analysis primarily concentrate on 8x8 asymmetric S-Boxes which are presently under wide use.

6.1 Random S-Box

In this section we generate and analyze 5 purely random S-Boxes. Values for each position were selected randomly so that the only intentional structure in the resulting s- box was that each S-Box contained values between 0 and 255 without repetition. The characteristics of these S-Boxes are basically recorded in order to observe the improvements achieved through the analytical approach or the conditional filling. Table 3 shows the test results that were obtained for the five random S-Boxes.

Table 3 Results of randomly generated S-Box

S-Box No	R	N	L	SAC
1	0.9375	0	16	7.1484
2	0.9453	0	14	8.9063
3	0.9375	0	16	6.2695
4	0.9375	0	16	6.6602
5	0.9531	0	12	6.5625

We see that the values of L are on an average close to 16 and the SAC values are below 30. Such an S-Box has an average resistance to differential cryptanalysis. More over the value of L is not consistent. The poor value of SAC indicates that it does not introduce much interdependency between the bits which is against the requirement of a good S-Box.

6.2 AES S-Box

Advanced Encryption Standard (AES) is a symmetric-key encryption standard adopted by the U.S government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each of these ciphers has a 128-bit block size, with key sizes of 128, 192and 256 bits, respectively. The AES ciphers have been analysed extensively and are now used worldwide, as was the case with its predecessor, [4] the Data Encryption Standard (DES). The AES encryption has four transformations namely the AddRoundKey, SubByte, ShiftRow and MixColumn [5]. In the SubByte transformation the algorithm makes use of 8x8 S-Boxes. Here each byte of the state is passed through the S-Box to obtain a new state which is then passes on to the next transformation. These 8x8 S-Boxes are generated using the Affine Transforms AT and AT^{-1} (Rijndael method) and are known for its high resistance to linear

and differential cryptanalysis. As a 'control' against which to compare the quality of the S-Boxes generated using our analytical methods, 5 AES S-Boxes were generated and tested. It was observed that all the five S-Boxes gave the same result for all tested parameters R, N, L and SAC. Thus only one set of values have been shown in Table 4.

Table 4 Analysis results of AES S-Box

S-Box No	R	N	L	SAC
1	0.9844	0	4	29.9805

From Table 4 it's seen that these S-Boxes have a less value of L=4 and a robustness R=0.9844 which is a good proof for its high resistance to differential cryptanalysis. The one problem we notice here is the low value of SAC which is near 30 and is almost comparable to the ones generated randomly (shown in Table 3). In an encryption algorithm it is more desirable if the degree to which the output bits are dependent on the input bits is higher. This dependency in introduced primarily through the S-Box and the efficiency of an S-Box in doing this is given by the value of SAC.
Strict Avalanche Criteria (SAC) requires that a change in one input bit results in the change of half the number of output bits. The ultimate goal is that every output bit of an encryption algorithm be dependent on every input bit. Such low values of SAC indicates that the S-Box is introducing less bit dependency and thus the burden of bringing about desirable bit dependency falls on other encryption steps.

6.3 Conditional Filling

Since the random generation of an S-Box inherited poor quality we try to find an optimum way to fill the S-Boxes which would induce better SAC and R. During the research we found that it was not only the conditions imposed that decided the quality, but it was also the filling pattern or algorithm that brought about a difference. A detailed generation and analysis of various fruitful filling algorithms using both the conditional hierarchies CH1 and CH2 have been discussed in this section. The reason behind testing all filling techniques with both the algorithm is that different algorithms worked well with different conditional hierarchy.
Note: all comparisons of improvement or decline in S-Box characteristics are made in comparison to the characteristics of the AES S-Boxes unless and otherwise stated. We start with the generation and analysis of S-Box using the Random Positioning algorithm and then proceed to Linear Filling and Neighbour First Filling algorithms.

6.3.1 Random Positioning

In order to achieve good robustness it is required that the generated S-Box is homogenous i.e. the elements satisfying a particular condition must be well spread throughout the S-Box and must not get concentrated in a particular region. This should be strictly satisfied particularly for the higher order conditions. It ensures that the value of L is kept low which it turn increases the robustness. To do this we adopt the random positioning algorithm which chooses an unfilled element from the S-Box and fills it with the best possible value. The S-Boxes generated using this algorithm when analysed give the properties shown in Table 5 and Table 6. The column named 'Time(s)' shows the time taken to generate the S-Box in seconds.

Table 5 Random Filling (CH1)

S-Box No	R	N	L	SAC	**Time(s)**
1	0.9609	0	10	43.75	150
2	0.9609	0	10	44.82	152
3	0.9531	0	12	45.80	142
4	0.9531	0	12	45.31	154
5	0.9531	0	12	44.34	145

The conditional hierarchy CH1 was used to generate these five S-Boxes. It's observed that they have an average robustness of 0.9562 which is 2.86% lesser as compared to AES S-Box and 1.5% better than the randomly generated S-Box. The effects of the conditions imposed are more evident when we look at SAC values. It's observed that the average value of SAC has gone up by 44.44% which is a positive sign.
Now we analyse the characteristics of S-Boxes generated using CH2. As discussed earlier the CH2 was formulated particularly to increase the value of SAC with a little compromise in R. The analysis results of these S-Boxes are shown in Table 6.

Table 6 Random Filling (CH2)

S-Box No	R	N	L	SAC	**Time(s)**
1	0.9531	0	12	48.54	77
2	0.9531	0	12	44.73	81
3	0.9531	0	12	45.02	75
4	0.9531	0	12	47.75	71
5	0.9531	0	12	45.21	81

As expected, the values of SAC have gone up (but only to a small extent) while the average robustness has decreased slightly to a value of 0.9531 as compared to the previous case shown in Table 5. Importantly the time taken for generating the S-Box has decreased by almost half, this is mainly due to the elimination of condition $3 \leq \alpha(2) \leq 5$ and $3 \leq \alpha(2) \leq 5$. In this algorithm the use of CH2 in place of CH1 has not given any appreciably improvement in the value of SAC, however from Table 5 and Table 6 it is notable that the imposed conditions CH1 and CH2 are having a great impact on the SAC values without much compromise on the values of R.

In Random Positioning algorithm there is no control over choosing the next position where the value is to be filled.

Due to this the first few randomly chosen places don't have any 1bit change neighbours and thus a randomly chosen unused value is filled into each of them. Until the S-Box is partially filled this problem keeps degrading the maximum possible quality that can be achieved. This basically is the prime reason which is holding back the SAC levels.To overcome this we device other filling algorithms, wherein the selection of place is controlled.

6.3.2 Linear Filling

The most basic and natural way of filling the S-Box is the linear way. Here the first cell (0, 0) is taken as the starting place and the S-Box is filled in a linear fashion from there. Since the next position to be filled is fixed and not random it makes this into a controlled filling technique. The analysis results using this algorithm are shown in Table 7 and Table 8 respectively.

Table 7 Linear Filling (CH1)

S-Box No	R	N	L	SAC	Time(s)
1	0.9375	0	16	70.80	97
2	0.9453	0	14	73.73	93
3	0.9375	0	16	71.78	95
4	0.9375	0	16	73.83	95
5	0.9375	0	16	72.07	96

The value of SAC in Table 7 has rocketed to an average of 72.44, which is highly desirable. It is more than twice the SAC value of AES S-Box. However the value of robustness has come down to approximately 0.9391 which is a 4.6% decrease. This is quite eclipsed by the huge increase in the SAC.
Performing the same using CH2 we expected to get a better value of SAC. The analyse show the following results as tabulated in Table 8.

Table 8 Linear Filling (CH2)

S-Box No	R	N	L	SAC	Time(s)
1	0.9453	0	14	75.98	49
2	0.9453	0	14	71.00	51
3	0.9453	0	14	75.56	44
4	0.9453	0	14	74.90	41
5	0.9453	0	14	75.10	47

Table 8 shows us a little more increase in the value of SAC as compared to Table 7 as expected. We also see that the value of robustness has gone up to an average of 0.9453 which was not predicted. These S-Boxes shows a 148% increase in the SAC value with a 3.97% decrease in R which is a great sign of overall improvement in quality. The decrease in the value of R can be characterised to the filling method. Since the algorithm goes about filling linearly the better values get accumulated in the upper half while the less desirable values are stacked at the bottom. This happens primarily due to the fact that lesser number of values is available to choose from by the time the filling reaches the bottom. However these S-Boxes can be used in cases where the SAC plays a major roll and it also serves as an example to show that how equally important the filling techniques are as when compared to the condition hierarchies. In some cases it is necessary to have a good value of robustness and the value SAC can be compromised to a certain extent. Such requirements are dictated wholly by the encryption algorithm, as for example the AES emphasis on having a greater value of R where as the DES gives an equal emphasis on both R and SAC. We proceed to discuss about the neighbourhood filling algorithm which produced the best balance between R and SAC among all the tested algorithms.

6.3.3 Neighbourhood Filling

As discussed earlier in table 8, by linear filling, the SAC value turns out to be high with a decreased robustness while the random filling has given a lower value of SAC with increased robustness. This algorithm which was formulated to strike a balance between the previous two algorithms is explained below.

Step1- At the start, a vacant space is chosen randomly.
Step2-The chosen place is filled with the best possible value. After filling, the position is pushed into a queue.
Step3-The queue is popped and all its 1bit change neighbours are found.
Step4-All these neighbours are filled one by one and each position is pushed into the queue after being filled.
Step5-If the S-Box is partially filled GoTo Step3. Step6-Stop
From the above steps we see that this algorithm goes on filling the neighbours and then the neighbours of neighbours and so on until the S-Box is completely filled. Using this algorithm ten S-Boxes were created and analysed. As earlier five of them were generated using CH1 and the rest using CH2. The analysis results obtained are shown in Table 9 and Table 10.

Table 9 Neighbours First Filling (CH1)

S-Box No	R	N	L	SAC	Time(s)
1	0.9609	0	10	48.05	113
2	0.9609	0	10	48.54	117
3	0.9531	0	12	45.51	124
4	0.9609	0	10	48.14	109
5	0.9609	0	10	44.82	129

These results show that the average value of SAC has increased by 56.95%. This increase is comparable to that achieved by random positioning. However the average value of robustness has showed a slight improvement and is also more consistent. Its average value is seen to be 0.9593 which is just 2.54% less as when compared to AES S-Box. Next we see the analysis results that were obtained when this algorithm was applied using the CH2 hierarchy. The results obtained are shown in Table 10.

Table 10 Neighbours First Filling (CH2)

S-Box	R	N	L	SAC	Time(s)

No					
1	0.9531	0	12	54.79	47
2	0.9609	0	10	54.00	65
3	0.9609	0	10	53.32	67
4	0.9531	0	12	54.59	59
5	0.9531	0	12	54.98	56

Finally, by neighbours first filling (CH2), we observe to get quiet an optimized values of robustness R=0.9561 and SAC=54.336. The value of SAC has increased by 81.24% and the robustness is comparable to those values in Table 9 and Table 6. We see that this algorithm gives a better value of SAC without any decrease in therobustness when compared to Random Positioning. This is primarily due to the elimination of the few degrading cycles which exists in the beginning of Random Filling where the vacant places being filled don't have any prefilled 1bit change neighbours. The elimination of such cycles is ensured by the algorithm through Step 3 and Step 4.The problem faced in Linear Filling i.e. the lack of spreading is also overcome by adopting to fill in the neighbours in every round.

VII. WIDE USABILITY

Since our approach of inducing quality into S-Boxes does not depend on any geometric factors such as size of S-Box or the number of input and output bits, it can be widely used for generating many kinds of S-Boxes. It can be a DES like S-Box which has a 6bit input and a 4bit output or a symmetric S-Box which is used in involution block ciphers or any other kind of S-Box which carries its own special characteristics. Our algorithm just involves a conditional hierarchy and filling technique both of which can be modified to generate S-Boxes of required behaviour.

As a support to our above statements, we generate and analyse briefly 8x8 'Symmetric' S-Boxes i.e. S-Boxes that satisfy the condition $S(S(x))=x$ using the same process which was used to generate the Asymmetric S-Boxes in the previous section. These symmetric S-Boxes play a major role in involutional block ciphers where the same algorithm is used for both encryption and decryption. These reversible algorithms come in handy when there is a resource constraint [6]. Since the S-Boxes are symmetric we have to fill in both the ends simultaneously. Thus both the ends have to be tested before filling up a place i.e. before filling the value '*y*' in position *x* it has to be tested whether the value '*x*" satisfies a good condition in position *y*. This is done by collecting all the *y"s(„y1","y2","y3"...."yn")* that are suitable to be filled in position *x* and then finding out the best position *y(* among *y1,y2,y3....yn)* where in the value '*x*' satisfies the best possible condition among all the alternatives.

Here too various conditional hierarchies were tested with different filling techniques. We discuss here only the two best combinations.

7.1 Linear Filling (CH2)

In this we filled in the S-Box linearly using the CH2 conditional hierarchy. Five S-Boxes were generated and analysed, the results are shown in Table 11.

Table 11 Linear Filling (CH2)

S-Box No	R	N	L	SAC	Time(s)
1	0.9453	0	14	49.22	16
2	0.9531	0	12	49.02	19
3	0.9531	0	12	44.04	16
4	0.9531	0	12	45.51	17
5	0.9375	0	16	48.82	17

Since these S-Boxes need to be symmetric it greatly reduces the freedom in choosing the values for a position. This results in a drastic decrease in the maximum value of SAC from 75% in asymmetric to 47% here. However the time taken to generate these S-Boxes is much less as compared to the time taken to generate an asymmetric S-Box. This is primarily due to the filling of two values in each step.

7.2 Alternative Filling (CH1)

Here the filling process has two cycles. It the first cycle it starts with the first element 0 and fills up all the even numbered positions. The second cycle starts with element 1 and fills up all the leftover odd numbered positions. By doing this we ensure that the usable values don't get exhausted in one portion of the S-Box. The first round fills up all the even values with good quality numbers consistently mainly due to the ensured availability. Five S-Boxes were created using this method and its analysis results are shown in Table12.

Table 12 Alternative Filling (CH1)

S-Box No	R	N	L	SAC	Time(s)
1	0.9531	0	12	42.38	57
2	0.9531	0	12	44.73	37
3	0.9531	0	12	44.36	44
4	0.9531	0	12	44.04	41
5	0.9531	0	12	42.06	44

The consistency in the value of R can be clearly observed. However this has come at the expense of a small decrease in the value of SAC as compared to the values in Table 11.

VIII. CONCLUSION

An investigation has been conducted into the generation of substitution boxes (S-Boxes) using an analytical techniques. They involve randomly choosing a value and then testing it against a set of criteria to determine if it is suitable for inclusion in the s- box. The objective was to generate S-Boxes that meet the strict avalanche criteria (SAC), are non-linear, and have a high degree of resistance to differential cryptanalysis. The main focus was on generating 8x8 asymmetric and symmetric

S-Boxes which are under wide use presently. It was found that the probability of obtaining an S-Box that fully complies with the SAC using these methods is very low. However, a method was found that produces asymmetric S-Boxes which exhibit up to 75% compliance with the SAC. The maximum value of robustness achieved was 0.9606 and controlling robustness was found to be considerably more difficult compared SAC. Although the potential influence of an individual element on S-Box behavior can be estimated when testing for SAC compliance, there is no easy way to determine the cumulative effects of all the elements during the construction process. Hence, although a large percentage of the elements may meet the SAC it is possible that undesirable characteristics may come about that are not detectable until the S-Box is tested. For example, there may be a large number of instances where if a particular input bit is changed the same 3-bit output change will result. Although it is desirable that a 1-bit input change results in a 3-bit output change, if enough of these input/output pairs group together in the same position in the DDT, it will have a detrimental effect on robustness. There are other desirable S-Box features that have not been considered here, many of which are application dependent. However this investigation has shown that it is possible to generate S-Boxes, in a random manner, which can meet desirable criteria to a high degree.

REFERENCES

[1] B. Schneier, Applied Cryptography, "Protocols, Algorithms, and Source Code in C", *2nd Ed., Wiley*, New York, 1996.

[2] M. Matsui, "Linear cryptanalysis method for DES cipher, in Advances in Cryptology" *Eurocrypt'93, Springer-Verlag Lecture Notes in Computer Science*, 765 (1994), pp 386–397.

[3] E. Biham and A. Shamir, Differential cryptanalysis of DES-like cryptosystems, in Advances in Cryptology" , *Crypto'90, Springer-Verlag Lecture Notes in Computer Science*, 537 (1991), pp 2–21.

[4] NIST reports measurable success of Advanced Encryption Standard.

[5] Edwin NC Mui "Practical Implementation of Rijndael S-Box Using Combinational Logic", 2007

[6] K. Chmiel, A. Grocholewska-Czurylo, J. Stoklosa "Involutional Block cipher for Limited Resources" *IEEE Global Telecommunication Conference*, 2008, pp 1-5.