A
Dissertation

On

Boolean Functions Suitable for Strong Cipher Systems

Submitted in partial fulfillment of the

Requirements for the award of the degree

Of

MASTER OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Submitted by
Sonia Malik
M.TECH (CS)-2nd year
Enrollment No. 14535046

Under the guidance of
Dr. Sugata Gangopadhyay
Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
INDIAN INSTITUTE OF TECHNOLOGY
ROORKEE – 247667

Candidate's Declaration

I declare that the work presented in this dissertation with the title **"Boolean Functions Suitable for Strong Cipher Systems"** towards the fulfilment of the requirements for the award of **Masters in Technology** in Computer Science and Engineering submitted in the Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, India, is an authentic record of my own work carried out during the period from **July 2015-May 2016** under the supervision of **Dr. Sugata Gangopadhyay**, Associate Professor, Department of Computer Science and Engineering, IIT Roorkee. The content of this dissertation has not been submitted by me for the award of any other degree of this or any other institute.

DATE:…………….. SIGNED…………………………..

PLACE: ROORKEE (SONIA MALIK)

Certificate

This is to certify that the statement made by the candidate is correct to the best of my knowledge and belief.

DATE:…………….. SIGNED…………………………..

(DR. SUGATA GANGOPADHYAY)

Associate Professor

Dept. of CSE, IIT Roorkee

Acknowledgements

I would like to take this opportunity to extend my heartfelt gratitude to my guide and mentor Dr. Sugata Gangopadhyay, Associate Professor, Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, for his trust in my work and his able guidance. He was a regular source of encouragement and assistance throughout this dissertation work. His wisdom, knowledge and commitment to the highest standards have always inspired and motivated me. He has been very generous in providing the necessary resources and guidance for me to carry out my research. He is an inspiring teacher, a great advisor and most importantly a nice person. On a personal note, I would like to say that I am forever indebted to my parents for everything they have done for me and everything they have given to me. I thank them for the sacrifices they made for me, so that I could grow up in a learning environment. They have stood by me in everything I have done, providing constant support, encouragement and love. I also owe a lot to my friends for their company and the time they spent helping me out whenever I needed them.

Abstract

Cryptography is one of the important areas in computer science. For strong cipher systems Boolean functions and s-boxes with required cryptographic properties need to be developed. Many attacks on cipher systems consist of approximating the component Boolean functions and the component s-boxes. Many unitary transforms are useful in analyzing the strength of these Boolean functions and the S-boxes against the approximation. A fast and efficient system was implemented to calculate Walsh Hadamard Transform and Nega Hadamard Transform which was further used to calculate non-linearity and PAR values. This system was further extended to calculate the HN transform set and related PAR values. A fast method to calculate the HN transform set is incorporated in the PAR calculation system and results are analyzed. An s-box is chosen and its PAR values are compared with standard s-boxes and results are analyzed. A new method incorporating the Hill Climbing heuristic is also implemented to analyze the use of Nega Hadamard Transform in finding Boolean functions with better non-linearity and results were found to be improved.

List of Contents

List of Tables:

List of Figures:

CHAPTER 1

INTRODUCTION

Cryptography is one of the most significant areas of computer science used in today’s world to secure data. Many cryptographic algorithms are used for providing security of information and communication system. The ability of any cipher system to resist cryptanalytic attacks depends on the individual components of which it is comprised. For e.g. Boolean functions and S-boxes or the substitution boxes are the most important and effective components of many cipher systems. A lot of research is ongoing on developing strong Boolean functions and the S-boxes having required cryptographic properties which are strong enough to resist cryptanalytic attacks.

Many heuristic methods have also been proposed to find Boolean functions with better cryptographic properties [1]. Differential cryptanalysis and the linear cryptanalysis are two main types of cryptanalytic attacks on block ciphers. Linear Cryptanalysis is one of the known plaintext attack. It is based on exploiting the characteristics of S-boxes. Each s-box is a combination of Boolean functions which have the property of non-linearity. So, if the non-linearity of these functions is very low then we can approximate these functions using linear expressions [6]. And in this way we can approximate the key used. So, to prevent linear cryptanalysis the s-boxes and Boolean functions in turn must be highly non-linear.

Linear cryptanalysis attack is used to find the key bits of a block cipher by making an approximation of each block cipher round by Z_2 linear expressions [6]. More generally the approximations can be done using linear expressions over any weighted alphabet. The Z_2 linear expressions approximating the block cipher rounds can be guessed by doing an analysis with respect to the Walsh Hadamard Transform of the constituent Boolean functions of S-boxes. Similarly Nega Hadamard Transform can also be used in the study of these generalized approximations. Some other linear unitary transforms like HN Transform are also useful in spectral analysis of Boolean functions and S-boxes in the study of these generalized approximations. These transforms also help us to evaluate the non-linearity of the constituent Boolean functions of an s-box along with analysis of approximation. So, a generalized study of these transforms is made in order to further assist the cryptanalysis and making these Boolean functions strong.

A faster way to calculate the Walsh Hadamard Transform was implemented for this purpose. Corresponding to this the fast Nega Hadamard Transform was also implemented for s-boxes. Similarly the HN transform set is difficult to be generated by brute force method. A method to calculate the corresponding entries of HN transform matrix is studied and implemented. So, a generalized system is implemented for calculating various transforms in an efficient way possible which can assist in evaluating the non-linearity aspect of s-boxes and approximating its constituent functions. This system is further used to calculate the non-linearity of a newly proposed s-box. The results helped us to choose the s-box as a reference for further study because its non-linearity was comparable to AES s-box.

S-boxes are an important part of any block cipher. So, a study of s-boxes of various modern block ciphers is also made along with a newly proposed s-box example. The non-linearity aspect of s-boxes can be quantified in terms of Peak to Average Ratio (PAR) [6] which can be calculated using various transforms. PAR can be further used to guess whether the generalized linear approximations of constituent Boolean functions can be done or not.

A system was implemented for calculating the PAR values of s-boxes. This system is further extended to calculate the PAR value for HN Transform set by incorporating the fast method for calculating the HN Transform set. The entries of any HN Transform matrix can be calculated using a formulae and this makes the PAR value calculation faster. From the analysis of results of PAR for s-boxes of various modern ciphers and the example s-box it can be shown that the Z_2 linear approximations for modern ciphers are difficult but generalized linear approximations can be done. Calculation of PAR values for HN Transform [7] set helped in showing that generalized linear approximations are more helpful. Also the example s-box has high PAR value which suggest that Z_2 linear approximations as well as generalized linear approximations for this S-box can be done more easily.

As it has been established that highly non-linear Boolean functions and S-boxes are suitable for strong cipher systems, many methods have been proposed for developing these. Many heuristic methods like Hill climbing are used to develop the Boolean functions gradually with better cryptographic properties. Walsh Hadamard Transform can be taken as one of the parameters to judge the non-linearity of Boolean functions in these methods. A new method is proposed incorporating the hill climbing method which takes in the Walsh Hadamard Transform to find the Boolean functions with better non-linearity. Nega Hadamard Transform can also be used to judge the non-linearity of Boolean functions along with some additional properties. So, the Nega Hadamard Transform is also included in the method to find the Boolean functions in a better way by avoiding the conditions of local maxima and jumping to a Boolean functions with better non-linearity. Improved results were obtained using the Nega Hadamard Transform in the method.

1.1 Background and Motivation

Cryptanalysis is generally used to find the key bits of a block cipher so that the cipher system can be cracked. Since Boolean functions and the S-boxes are the constituent parts of a cipher system, the cryptanalysis is used to approximate these Boolean functions and the S-boxes. Generally the approximation of these Boolean functions and S-boxes is Z_2 –linear. But this can also be generalized and can be taken over any Z like Z_2 as long as the approximation is linear in a way that it can be formed from tensor product of length 2 vectors. The approximations can be easily found using the spectral analysis of Boolean function with respect to various transforms.

A system of various linear unitary transforms have been developed including Walsh Hadamard Transform, Nega Hadamard Transform and HN Transform. A fast method for calculating all of these transforms need to be developed for such an analysis. A fast method for calculating Walsh Hadamard Transform is already available but there is a need to develop fast method for other transforms as well because the brute force method for calculating the other transforms is exponential. For example the brute force method to calculate the Walsh Hadamard Transform takes $O(2^{2n})$ which is only after leaving the time to calculate the Walsh Hadamard Transform Matrix which is also a recursive method. Similarly, the HN Transform is a set of 2^n matrices each of which takes an exponential time to be calculated by the tensor products of length 2 vectors [7]. A method is needed to be developed to easily calculate these 2^n matrices.

For the analysis of Boolean functions and the s-boxes a lot parameters are already available like non-linearity etc. But other parameters like Peak to Average Power Ratio are also helpful in analysis against approximations. A system is need to be developed to calculate PAR values for various transforms and this can be used to test whether new s-boxes are suitable for any cipher system. This system also need to be fast so that large s-boxes can also be analyzed.

A lot of methods for generating various Boolean functions and s-boxes are already available like brute force search and heuristic methods like hill climbing etc. In Hill Climbing any cryptographic property can be used to modify a Boolean function truth table to reach to a Boolean functions with better cryptographic properties [12]. This method is proved to be a very good method for generating the better Boolean function. Since Nega Hadamard Transform is a recent development in the field of Linear unitary transforms and PAR is a new parameter to judge the various transforms we can propose a new method incorporating the Hill climbing to generate better Boolean functions. There are some problems in the methods of Hill climbing like the local maxima where the Boolean function has the best possible value of the desired cryptography as compared to nearby Boolean functions but not overall. A property of the Nega Hadamard Transform can be useful to solve this problem. So a method can be proposed incorporating the Hill climbing method and the PAR values of Walsh Hadamard Transform spectrum and the Nega Hadamard Transform.

1.2 Objectives and Outcomes

The following are the objectives and outcomes of the work presented in this report

a. To implement a system for calculating the various transforms in a faster and efficient way. This system can be further used to calculate the non-linearity aspect of s-boxes and also in the linear cryptanalysis of block ciphers. Walsh Hadamard Spectrum and Nega Hadamard Spectrum has been calculated in this system. This system is further extended to calculate the HN Transform matrices and corresponding spectrum.
b. To select an example s-box and calculate its non-linearity by using the above system. Its non-linearity should be comparable to the AES s-box's non-linearity so that this s-box can be further analyzed with respect to AES s-box.
c. To understand the importance of various transforms and Peak to Average Ratio (PAR) value in linear cryptanalysis of a block cipher and to implement a system for calculating the PAR for various transforms and analyze the results for various s-boxes and the newly selected s-box. This system is extended to calculate the PAR values for the HN Transform set and results are analyzed for various s-boxes.
d. To develop and implement a new method incorporating the hill climbing heuristic and PAR values analyzing the use of Nega Hadamard Transform in finding Boolean functions with better non-linearity.

1.3 Structure of the report

The report has been divided into 4 sections including this introduction section. Section 1 describes the introduction which consist of the basic overview of the entire work done in the thesis. The background of the thesis work and the motivation for the new work done in the thesis is also described in this section. The objectives and outcomes of the thesis are also described in this section which describes the problem statement and the proposed solution of the thesis in brief. Section 2 will describe the basic terminology related to further work in following sections. This section also describes the previous work done and background knowledge required to understand the further work. This includes definitions of various transforms like Walsh Hadamard Transform, Nega Hadamard Transform, HN Transform, linear cryptanalysis, Peak to average Power Ratio i.e. PAR values, Heuristic methods like Hill climbing etc. Section 3 will consist of the Methodology followed for achieving all the objectives mentioned above and will also consist of the results obtained and their analysis. Section 4 will describe conclusion and future work.

CHAPTER 2 : BASIC TERMINOLOGY

2.1 Boolean function and S-boxes

An n-variable Boolean function f(x) is an association from vector space of n-dimensions over F_2 to F_2, where F_2 is a field corresponding to {0, 1}. For example a 3-variable Boolean function f(x) can be a function which maps 3-digit binary numbers to 1-digit binary number {0, 1}. The truth table of a Boolean function is a vector corresponding to the outputs of the Boolean function for all the inputs taken in lexicographical order. The polarity truth table of a Boolean function denoted by $\hat{f}(x)$ is a truth table corresponding to the values $(-1)^{f(x)}$ where the input values of x are taken in lexicographical order [1]. The hamming weight of a Boolean function f(x) is the number of ones in the output vector i.e. truth table.

S-boxes are nothing but multiple output Boolean functions. Various cipher systems like block ciphers make use of S-boxes. To study the S-boxes the constituent Boolean functions are analyzed. Suppose a given S-box has n inputs and m outputs. This means that our S-box consist of m Boolean functions each of n inputs. While analyzing the S-box, we not only consider the constituent Boolean functions but also consider all the functions of the form [10]

$$f(x) = \sum_{i=0}^{m-1} c_i f_i(x) \tag{2.1}$$

Where f are the constituent Boolean functions and c_i belongs to Z_2

2.2 Walsh Hadamard Transform

Walsh Hadamard Transform is a kind of a function which can be calculated for any Boolean function. It is a useful transformation which can be used in evaluating various cryptographic properties of a Boolean function like non-linearity. It can also be used in linear cryptanalysis of Boolean functions and S-boxes. Walsh Hadamard Transform of a Boolean function f on V_n (the values of f are real numbers 0 and 1) is the mapping W (f): V_n -> R, defined by [1]

$$\text{W (f) (w)} = \sum_{x\in V_n}(-1)^{f(x)}(-1)^{w.x} \tag{2.2}$$

Here w.x is the scalar or dot product of the Boolean variables w and x and it is defined as follows:

$$w.x = w_n x_n \oplus w_{n-1} x_{n-1} \oplus \ldots . . \oplus w_1 x_1 \qquad (2.3)$$

Here $\oplus$ denotes the addition over Z_2^n or simple the XOR.

Walsh spectrum of a Boolean function f is the list of the 2^n Walsh coefficients corresponding to the 2^n values of w. Walsh Hadamard Spectrum can also be calculated using Hadamard matrix [2] instead of the formulae used above. Using this method first of all the Hadamard matrix of size $2^n \times 2^n$ is calculated recursively for a Boolean function of n variable size. Then the hadamard matrix is multiplied with the Boolean function taken as a column matrix and the hadamard spectrum is obtained.

2.3 Kronecker product

Kronecker product can be defined as one of the binary operation between two matrices. It is denoted by the symbol $\otimes$ i.e. A $\otimes$ B [2][6]. Basically in this operation the second matrix overlays or occupy each of the positions of the first matrix multiplied by each element of the first matrix. It can be easily shown with the help of an example. For e.g.

$$A = \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 5 & 8 \\ 6 & 7 \end{bmatrix} \qquad (2.4)$$

Then

$$A \otimes B = \begin{bmatrix} 1.B & 4.B \\ 2.B & 3.B \end{bmatrix} \qquad (2.5)$$

i.e.

$$A \otimes B = \begin{bmatrix} 1.5 & 1.8 & 4.5 & 4.8 \\ 1.6 & 1.7 & 4.6 & 4.7 \\ 2.5 & 2.8 & 3.5 & 3.8 \\ 2.6 & 2.7 & 3.6 & 3.7 \end{bmatrix} \qquad (2.6)$$

Here '.' Is simply the multiplication.

Kronecker product is associative in nature. i.e.

$$A \otimes B \otimes C = (A \otimes B) \otimes C = A \otimes (B \otimes C) \qquad (2.7)$$

2.4 Hadamard matrix

H of order n is an n×n matrix of ±1 s such that $H_0 = 1 \quad and \quad H_1 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

$$H_n = \begin{pmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{pmatrix} \tag{2.8}$$

$$H_n = H_1 \otimes H_{n-1} \tag{2.9}$$

where ⊗ is the Kronecker product which is an operation between 2 matrices such that the second matrix is multiplied with the values of first matrix and overlaid onto the first matrix. For e.g. in eqn. (2.8) H_{n-1} values are multiplied with H_1 values[2][6].

Using this method we can calculate the Walsh Hadamard Transform by first calculating Hadamard Matrix and then multiplying the function with that matrix. The Hadamard matrix can be calculated using a recursive method by using the above formulae step by step. i.e. by first calculating H_2 then H_3 then H_{n-1} and soon then H_n. Then the function is taken as a column matrix and the Hadamard matrix is multiplied with the function to get the Walsh Hadamard transform. It will take O (2^{2n}) where n is the number of input variables of f(x) to calculate the Walsh Hadamard Spectrum using this method. This can be considered as a slow method of calculating Walsh Hadamard spectrum as the time taken is exponential. So a fast method need to be developed for calculating Walsh Hadamard Spectrum.

2.5 Fast Walsh Hadamard Transform

The Sylvester Hadamard Matrix H_n can be decomposed as [3]

$$H_n = {M_n}^{(1)}{M_n}^{(2)} \ldots\ldots {M_n}^{(n)} \tag{2.10}$$

Where

$${M_n}^{(i)} = I_{2^{n-i}} \otimes H_1 \otimes I_{2^{i-1}} \tag{2.11}$$

I_m is the m×m Identity matrix. In this way we will find that we can broke down the process of multiplication of the function directly with the Hadamard matrix into n steps. In each of these n steps we will take into consideration M_n and can find a fast method to calculate the Walsh Hadamard spectrum with lesser time complexity. The details of this method will be described in the methodology section. We can calculate the walsh hadamard transform using this method in O(n 2^n) which is way faster than the matrix multiplication method.

2.6 Nega *Hadamard Transform*

Nega Hadamard Transform is another useful transform similar to WHT which was first defined by M.G.Parker. Here the Nega Hadamard Transform of a function f on Vn is a mapping from V_n ->C. where C is the complex set of numbers.i.e.[2]

$$N(f)(w) = \sum_{x\in V_n}(-1)^{f(x)}(-1)^{w.x}i^{wt(x)} \tag{2.12}$$

Where wt(x) is the weight of x which can be calculated by number of 1's in x. and w.x is the scalar product of w and x Boolean variables as defined in eqn. (2.3). The Nega Hadamard Spectrum is the list of all 2^n values corresponding to all 2^n values of w. it can also be calculated using the Nega Hadamard matrix which can be calculated using the Kronecker product. The Nega Hadamard matrix used here is

$$N_1 = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} \tag{2.13}$$

$$N_n = \begin{pmatrix} N_{n-1} & i * N_{n-1} \\ N_{n-1} & -i * N_{n-1} \end{pmatrix} \tag{2.14}$$

The Nega Hadamard matrix can be calculated recursively by the Kronecker product. Then the Nega Hadamard matrix can be multiplied with the Boolean function taken as a column matrix to get the Nega Hadamard Spectrum similar to the Walsh Hadamard Transform matrix. Using this method it will take O(2^{2n}) which is exponential. We can also calculate using fast method similar to WHT. So, a fast method need to be developed and an algorithm similar to Walsh Hadamard Transform can be generated to see if that will work. Since the calculation of various spectrums is useful in the analysis of Boolean functions

We have implemented both the fast method and slower method of WHT to compare them and also for further use in linear cryptanalysis. We have also implemented fast Nega Hadamard

Spectrum calculation similar to WHT and showed that it will also work for Nega Hadamard Spectrum and has a lower time complexity than the Nega Hadamard matrix method.

2.7 Tensor Linear Sequence

A sequence of length N which can be tensor decomposed according to the factors of N completely is called a tensor linear sequence[7]. For e.g. if N=2^n then the tensor linear sequence can be written

$$(a_0, b_0) \otimes (a_1, b_1) \otimes \ldots\ldots \otimes (a_{n-1}, b_{n-1}) \tag{2.15}$$

This definition is useful in terms of calculating transforms. For e.g. WHT can be written as a tensor linear sequence. Similarly other transforms can also be defined. For e.g. Walsh Hadamard Transform consist of all Tensor Linear Sequence which can be described in a form as follows:

$$(\pm 1, \pm 1) \otimes (\pm 1, \pm 1) \otimes \ldots\ldots \otimes (\pm 1, \pm 1) \tag{2.16}$$

For example 4×4 Walsh Hadamard Transform matrix can be written as

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{2.17}$$

$$= \begin{bmatrix} (1,1) & \otimes & (1,1) \\ (1,-1) & \otimes & (1,1) \\ (1,1) & \otimes & (1,-1) \\ (1,-1) & \otimes & (1,-1) \end{bmatrix} \tag{2.18}$$

2.8HN transform set

The HN transform set is also another linear unitary Transform which is denoted by $\{H, N\}^n$.[7] It is also useful in the linear cryptanalysis and approximation analysis of S-boxes and Boolean functions of various cipher systems like the Walsh Hadamard Transform and the Nega Hadamard Transform. Unlike other transforms like WHT and Nega Hadamard Transform, this transform doesn't consist of a single Hadamard matrix. Instead it is a set of 2^n matrices for calculating the transform for any n variable Boolean function. These matrices are obtained by taking the tensor products of the Hadamard kernel and the Nega Hadamard kernel[7]. The Hadamard kernel is denoted by H and the Nega Hadamard kernel is denoted by N and the identity transform is denoted by I.

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.19}$$

$$N = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} \tag{2.20}$$

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.21}$$

By taking the tensor product of H and N, n times in any sequence gives us 2^n different transform matrices which are linear unitary in sense. These are all denoted by $\{H, N\}^n$.

Let the n positions be denoted by numbers from 1 to n and let P_H and P_N denote the partition of {1….n} positions such that P_H denote the positions where Hadamard kernel is placed in the tensor product sequence and P_N denote the positions where Nega Hadamard kernel is placed. Then the Unitary transform matrix corresponding to this position distribution is given by[7]

$$U = \prod_{j\in P_H} H_j \prod_{j\in P_N} N_j \tag{2.22}$$

Here $X_j = I \otimes I \otimes I \ldots\ldots I \otimes X \otimes I \otimes I$

Where X is in the jth position in {1…n} positions. The spectrum of any function f can be calculated from multiplication of the matrix U with the column matrix of function taken in the polar form. The 2^n matrices consist of two special cases. i.e when P_H consist of all positions {1…n} then the U matrix is called the Hadamard matrix and the spectrum obtained by the multiplication with the function is called Walsh Hadamard Spectrum. Similarly when P_N consist of all the positions from {1…n} then the U matrix obtained is called the Nega Hadamard matrix and the spectrum obtained by the multiplication with the function is called Nega Hadamard Spectrum. If the absolute value of each output in the spectrum obtained by the multiplication of unitary matrix U and the polar values of function is 1 then the spectrum is called as flat spectrum. If any of the spectrum in the 2^n HN Transform set is flat then the function is called as Bent_4 function.

2.9Homogeneous symmetric Boolean function

A homogeneous symmetric Boolean function can defined with respect to any algebraic degree. It can be described in the Algebraic Normal Form which is a polynomial representation of a Boolean function. A Boolean function f(x) can be represented in the form[2]

$$f(x) = \sum_{a \epsilon V_n} c_a {x_1}^{a_1} \dots {x_n}^{a_n} \tag{2.23}$$

where $c_a \epsilon F_2$ and $a = (a_1, \dots \dots, a_n)$. Also, $c_a = \sum_{x \leq a} f(x)$ where x≤ a means that $x_i \leq a_i$, for all 1 ≤ i ≤ n. The algebraic degree of a Boolean function is the number of variables (x_i) in the highest product term having non-zero coefficient.

Let S_r (x) be any homogeneous symmetric Boolean function and its algebraic degree is r. Then it can be described in its ANF form as [7]

$$S_r(x) = \underset{1 \leq i_1 < \cdots < i_r \leq n}{\otimes} x_{i_1} \dots \dots x_{i_r} \tag{2.24}$$

For example let S_2(x) is a homogeneous symmetric Boolean function of algebraic degree 2 and has 3 variables then the S_2(x) can be written as

$$S_2(x) = x_1 x_2 \oplus x_2 x_3 \oplus x_1 x_3 \tag{2.25}$$

Similarly if S_3 (x) is a homogeneous symmetric Boolean function of algebraic degree 3 and 4 varibles then S_3 (x) can be written as

$$S_3(x) = x_1 x_2 x_3 \oplus x_1 x_3 x_4 \oplus x_1 x_2 x_4 \oplus x_2 x_3 x_4 \tag{2.26}$$

A system is also implemented in the thesis to calculate symmetric homogeneous Boolean function to be used in the proposed method incorporating the hill climbing method.

Another important calculation to be needed in the thesis work related to this is the intersection of two Boolean vectors. Let a = (a_n……a_1) and b = (b_n……..b_1) be two Boolean vectors belonging to Z_2^n. Then intersection of these two vectors is defined as a*b which is

$$a * b = (a_n b_n, \dots \dots \dots, a_1 b_1) \tag{2.27}$$

Then the symmetric function $S_r(a*b)$ is

$$S_r(a * b) = \bigotimes_{1 \le i_1 < \cdots < i_r \le n} \left(a_{i_1} b_{i_1}\right) \dots \dots \dots (a_{i_r} b_{i_r}) \tag{2.28}$$

2.10 Calculation of HN Transform set

As we have seen in the previous section that there are 2^n transform matrices in the set of HN Transform set for a Boolean function of n variables. And to calculate this we will have to perform the tensor products n times for each of the 2^n matrices which is a lot of computation. The time taken to calculate the HN transform spectrum will be exponential and includes 2^n times the calculation of each transform matrix which again takes exponential time because each matrix is calculated recursively. We can decrease the time complexity if instead of calculating the 2^n matrices recursively we get to know a way to calculate each entry of each matrix in a constant time. Then the time taken will be really $O(2^{2n})$ to calculate each matrix since the matrix is $2^n \times 2^n$ and to calculate each entry it takes only constant time.

Let the n positions be denoted by numbers from 1 to n and let P_H and P_N denote the partition of {1….n} positions such that P_H denote the positions where Hadamard kernel is placed in the tensor product sequence and P_N denote the positions where Nega Hadamard kernel is placed. Then the Unitary transform matrix corresponding to this position distribution is given by U and for any $, b \in Z_2^n$, the entry in the position a th row and b th column of $2^{n/2}U$ is[7]

$$(-1)^{a.b \oplus s_2(c*b)} i^{c.b} \tag{2.29}$$

Here $c = (c_n, \dots \dots \dots c_1) \in Z_2^n$ is assigned such that $c_i = 0$ if $i \in P_H$ and $c_i = 1 \; if \; i \in P_N$.
For example let a function be of 2 variables. Then n = 2. Then according to the above definition if c = (0,0) then U = H ⊗ H. similarly if c = (0,1) then U = H ⊗ N . if c = (1,0) then U= N ⊗ H and last if c = (1,1) then U = N ⊗ N. if we calculate all the entries using the above equ. (2.29) then we will get the correct unitary transform matrices[7]. For example below is the U

When c=(0,1)

$$H \otimes N = \frac{1}{2}\begin{bmatrix} 1 & i & 1 & i \\ 1 & -i & 1 & -i \\ 1 & i & -1 & -i \\ 1 & -i & -1 & i \end{bmatrix} \qquad (2.30)$$

When c=(1,0)

$$N \otimes H = \frac{1}{2}\begin{bmatrix} 1 & 1 & i & i \\ 1 & -1 & i & -i \\ 1 & 1 & -i & -i \\ 1 & -1 & -i & i \end{bmatrix} \qquad (2.31)$$

When c=(0,0)

$$H \otimes H = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \qquad (2.32)$$

When c=(1,1)

$$N \otimes N = \frac{1}{2}\begin{bmatrix} 1 & i & i & -1 \\ 1 & -i & i & 1 \\ 1 & i & -i & 1 \\ 1 & -i & -i & -1 \end{bmatrix} \qquad (2.33)$$

In this case if we check the entries according the formulae then it proves to be right. For e.g. here c = (0,1) and let a th row is 3 then a = (1, 0) and b th column is 3 then b = (1,0)
Then the entry will be

$$(-1)^{(1,0).(1,0)\oplus(0.1).(1.0)} i^{(0,1).(1,0)} = -1 \qquad (2.34)$$

Which is the same as the entry in the matrix shown in eqn. (2.30).

2.11 Linear Cryptanalysis using PAR

Block cipher uses a secret key and an algorithm consisting of many rounds to convert plaintext into cipher text. So, linear cryptanalysis is used to get the secret key by trying to approximate the core rounds of the block cipher which can further be combined to approximate the key bits. This approximation is usually made using Z_2 linear expressions for each round which can relate some input and output bits of each round with some probability. Then by using a lot of plaintext and cipher text pairs the probability of approximation to be correct is determined and the key bits are guessed accordingly [6] [8].

To prevent such attacks form being successful many block ciphers try to make the constituent Boolean functions highly non-linear so that they can't be approximated using Z_2 linear expressions. Let S be an n×n s-box of a block cipher and let f and g are the linear combinations of n input variables (x) and n output variables(y) respectively[6]. And let A be a subset of s-box i.e.all (x,y) pairs of the s-box. Then the cipher is said to be resistant to linear cryptanalysis over Z_2 if

$$f(x)=g(y) \text{ for all } (x,y)\epsilon\, A \tag{2.35}$$

$$\left||A| - 2^{n-1}\right| \leq 2^{\frac{n}{2}} \; if\; n\; is\; even\; and \left||A| - 2^{n-1}\right| \leq 2^{\frac{n-1}{2}} \; if\; n\; is\; odd \tag{2.36}$$

Most of the modern block ciphers make the Z_2 linear approximations difficult. The Z_2 linear approximations can be found with the help of spectral analysis of WHT. But we can also do the linear approximations of the constituent Boolean functions of an S-box[5] with respect to any weighted alphabet. Then we can combine these generalized linear approximations to retrieve the key bits. The way to combine these generalized linear approximations is for future research. We can then see that even these modern block ciphers are a little weaker with respect to these generalized linear approximations.

2.12 Peak-to-Average Power Ratio(PAR)

Peak to average power ratio (PAR) can be defined as one of the measures or criteria which can help us in analyzing the non-linearity and linear approximation capability of various Boolean functions and s-boxes. It is actually helpful in quantifying the non-linearity of various S-boxes and constituent Boolean functions in various cipher systems. It is calculated with respect to various Transforms like Walsh Hadamard Transform, Nega Hadamard Transform and the HN Transform set.

The WHT can also be defined as

$$\text{W (f) (w)} = 2^{-n} \sum_{x\in V_n} (-1)^{f(x)} (-1)^{w.x} \tag{2.37}$$

Then we can define PAR with respect to WHT as

$$\text{PAR}(f) = 2^n max_{\forall w}(|W(f)(w)|)^2 \tag{2.38}$$

If the value of PAR is 1 this means that no linear approximation is possible whereas 2^n means that the function is a linear function completely and can be easily approximated by a linear function. This means that higher the value of PAR the higher the chances of linear approximations. This is all w.r.t. WHT. PAR can also be defined for any normalized Transforms set T[6]. i.e.

$$PAR(f) = 2^n max_{\forall k \forall U \in T}(|F_k|)^2 \tag{2.39}$$

Here also it helps in finding the approximation possibility as the higher the value of PAR, more are the chances of better approximations. Using PAR the non-linearity of S-boxes of Block ciphers can be easily judged. By calculating the PAR of various S-boxes we can see and compare them to know which can be easily approximated or not. We can also know whether linear approximations over Z_2 are easier to do or over some other generalized set are easier to do. For this the highest and lowest PARs are calculated for different sets of transform and compared accordingly for different S-boxes.

S-boxes are multiple output Boolean functions and the transforms are calculated for Boolean functions basically. So to calculate the different transforms of S-boxes the various constituent Boolean functions of S-boxes are combined in the following way and then the transforms are calculated for each of them and accordingly the PAR is calculated. i.e.

Let the S-box has n inputs and m outputs. Then the constituent Booleans functions are

$$y_j = f_j(x) \quad , \quad 0 \leq j < m \ , f_j : Z_2^n \ \rightarrow \ Z_2 \tag{2.40}$$

And we can calculate the transform and PAR over all f:$Z_2^n \rightarrow Z_2$ of the following form,

$$f(x) = \sum_{i=0}^{m-1} c_i f_i(x) \ , \qquad\qquad c_i \in Z_2 \tag{2.41}$$

Here the f(x) will consist of 2^m Boolean functions of the form described in the eqn. (2.40). The largest PAR and smallest PAR both can then be calculated for the s-box. The largest PAR is calculated by first calculating the spectrum for each of these function f(x), finding the maximum value of transform from the spectrum for each of these functions , then taking the maximum out of these maximum values and then squaring and normalizing it. Whereas the smallest PAR is calculated by first calculating the spectrum for each of these functions f(x) , finding the maximum value of the transform from the spectrum for each of these functions and then taking the minimum out of these maximum values and then squaring and normalizing it. Although while calculating these values we will ignore the function f(x) which is formed by all c_i's taken as 0.

2.13 Heuristic methods

A lot of cipher systems make use of the Boolean functions and the S-boxes. To be suitable for a strong cipher system Boolean functions need to be having strong cryptographic properties. A lot of methods have been developed to create strong Boolean functions which have suitable cryptographic properties like non-linearity and auto correlation. Brute force method checks all the Boolean functions one by one and compares the cryptographic properties in consideration. But this method will not work accurately and fast in Boolean functions with large number of variables. One of the other best method to develop Boolean functions with strong cryptographic properties is the heuristic method. This type of methods are generalized by the search of the suitable Boolean function in a localized area directed by certain conditions and starting from specified points in the space.

Many heuristic method have been discovered till now. For e.g. Hill Climbing, genetic algorithms, simulated annealing etc. The basic logic in the process of hill climbing is to start from a specified point, modify the Boolean function one variable at a time only if there is an improvement in the original cryptographic value. That is why it is called hill climbing method i.e. climbing towards a better cryptographic function through search. The Boolean function obtained in the last is expected to be having the best cryptographic property.

Since PAR values can be used to quantify the non-linearity of Boolean functions and the s-boxes, we can use PAR values in the heuristic methods like Hill Climbing method[12] to test whether we will get better Boolean functions. Also it can be proposed that the PAR values with respect to the Walsh Hadamard Transform and the Nega Hadamard Transform can be helpful in the conditions of local maxima where the cryptographic property has the best value nearby but it may not be the best in the whole search space. A method is proposed incorporating the Hill Climbing method and PAR values to improve the heuristic method and test whether the PAR value give additional advantage over the usual non-linearity measure.

There are various conditions also which are taken into consideration whether to change a parameter of a Boolean function or not in a heuristic method like Hill Climbing[12]. This condition can be weak or strong. A strong condition always require that the cryptographic property value always increase while moving to next step in the search method whereas the weak property requires that the cryptographic property value may increase or remain same while moving to the next step in the search method. So Hill climbing method is chosen and modified to propose a method taking the PAR values of WHT and Nega Hadamard transform to see if better Boolean functions can be searched with better PAR values and local maxima problem can be resolved or not.

CHAPTER 3
METHODOLOGY AND RESULTS

This section consist of the methodology used to achieve all the four objectives mentioned above and the results obtained. With respect to the four objectives described above this section is also divided into 4 subsections.

3.1 WHT, Nega Hadamard Transform and HN transform calculation

A system was implemented to calculate the WHT and the Nega Hadamard transform which can be further used for calculation of non-linearity or the PAR values. The WHT calculation system is implemented by two methods: one by using Hadamard matrix and the other by using the fast method and the time taken by both the methods is compared for variable lengths Boolean functions[9][10].

The Nega Hadamard transform calculation is implemented only by the fast method which is designed using the same concept as is used for WHT.

The first method for calculating WHT is based on the following steps:

a. The Hadamard matrix is calculated recursively by using the H_1 matrix and kronecker product according to the size of the Boolean function taken as input.
b. The Hadamard matrix is then multiplied with the Boolean function taken as a column vector and the resultant column matrix is the Walsh Hadamard spectrum. The time taken by this method is noted. It has a complexity of O (N^2) where N is the size of input Boolean function.

The fast method for calculating WHT is a kind of a divide and conquer method based on the following steps:

a. The input Boolean function (output of Boolean function ordered lexicographically) is divided into 2 halves. The output result is also divided into 2 halves and these are calculated as follows: first output half is calculated by adding corresponding 2 input half elements(for e.g. a[0] added with a[n/2],a[1] is added with a[n/2+1] and so on.) and second half is calculated by subtracting 2 corresponding 2 half elements.

b. The output result is divided into 2 halves and the above step is repeated for each of them.
c. This process continues logN times until each half contain 1 element.

This method has a time complexity of O(NlogN). The fast method approximately take 0 seconds and is much faster than Hadamard matrix method. So, we implemented the Nega Hadamard Transform using only fast method.

Nega Hadamard Transform calculation is implemented using the the fast method similar to the fast Walsh Hadamard Transform method with a few modifications. The input Boolean function (in the form of truth table ordered lexicographically) is now represented as a complex number or we can say as a pair i.e. one with real part and one with the imaginary part. The input Boolean function's values are multiplied with the term $i^{wt(x)}$ where wt(x) is the number of ones in the input x of the Boolean function f(x). Then this term is represented as a complex number i.e. the $i^{wt(x)}$ will finally result in either 1, -1, i or –i and after multiplication with 1 or -1 this will result in a complex number. Then the same fast method is applied as was used for WHT but now the addition and subtraction is in terms of complex numbers. This method will calculate the Nega Hadamard Transform with a time complexity of O(NlogN).

The HN Transform calculation method consist of first calculating the 2^n transform matrices in the HN Transform set and then calculating the HN Transform spectrum by multiplying the matrices with the function in the polar form.

The brute force method for calculating the HN Transform set will consist of recursively calculating the 2^n matrices each by taking the tensor products of the Hadamard kernel and the Nega Hadamard kernel. This process will take exponential time for calculating each of the 2^n transform matrices. So this method is not implemented to calculate the HN Transform matrices and spectrum.

The faster method of calculating the HN Transform matrices consist of the use of the eqn. (2.27) to calculate each of the entry of 2^n matrices in a constant time[7]. The method to calculate HN Transform matrices in a faster way consist of the following steps:

a. Take the number of variables in the Boolean function whose HN Transform spectrum is to be calculated.
b. For each of the matrices from 0 to 2^{n-1}, c value is calculated.
c. Then for each of the c value or for each of the matrices, each position is calculated by using the formulae in the eqn. (2.29).

d. The output of all the 2^n matrices are stored in an output file.

The time complexity of this method to calculate HN Transform matrices as analyzed is found to be $O(2^n.2^n)$ which is $O(2^{2n})$.

For example for the calculation of PAR value of HN transform spectrum for the DES S-box, we need to calculate the HN Transform matrices for n=6 number of variables which will produce 64 large matrices of size 64×64. The time taken by the above method to calculate the HN Transform matrices for different number of variables is shown in the Table 3.1 below.

Table 3.1: Time taken to calculate HN Transform matrices

No. of variables	Size of HN Transform matrix	Number of HN Transform matrices	Time taken by fast method to calculate all matrices(in sec)
4	16×16	16	0.002
6	64×64	64	0.104
8	256×256	256	8.496
9	512×512	512	73.937

It can be seen from the Table 3.1 that the fast method for calculating the HN Transform matrices takes exponential time as analyzed theoretically i.e. $O(2^{2n})$. The slow method to calculate the HN Transform matrices is exponential times slower than this method and can be seen as hard to implement. So we have proposed a faster method to calculate the HN Transform matrices.

The HN Transform spectrum can be calculated using the matrices one by one and multiplying it with the column vector of polar form of Boolean function. The following steps show the algorithm:

a. The input Boolean function is taken in polar form and converted into a complex number with the imaginary part as 0. It is done since the HN Transform matrices consist of complex entries too.

b. Then the matrix multiplication is done taking into consideration the complex entries of the HN Transform which can be 1,-1,i,-i only. The resulting spectrum is also in the form of complex numbers.

3.2 Selection of an example S-box

An S-box is selected which was constructed using a particular type of fractional linear transformation. It is analyzed with respect to AES S-box by comparing its non-linearity with the non-linearity of AES s-box[1]. the selected S-box is a 16×16 S-box similar to the AES s-box[4]. The selected S-box is given in Table 3.2.

Table 3.2: the selected 16×16 S-box

221	69	158	6	34	81	146	193	241	242	240	0	182	217	10	45
206	153	74	21	154	54	173	73	251	110	117	231	63	84	143	164
151	236	246	76	70	98	129	157	28	204	23	199	49	220	7	178
160	96	131	67	75	127	100	152	82	254	228	145	65	196	31	162
194	126	101	33	106	130	97	121	78	189	38	149	137	68	159	90
92	50	177	135	174	255	227	53	138	181	46	89	32	55	172	195
218	223	4	9	52	39	188	175	119	102	125	108	156	40	187	71
80	3	224	147	213	165	62	14	198	47	180	29	19	86	141	208
120	134	93	107	216	43	184	11	226	66	161	1	114	212	15	113
186	64	163	41	252	91	136	230	133	229	253	94	72	237	245	155
20	2	225	207	118	179	48	109	22	132	95	205	42	5	222	185
192	238	244	35	77	197	30	150	170	111	116	57	124	37	190	103
26	36	191	201	105	85	142	122	171	8	219	56	176	27	200	51
167	24	203	60	144	99	128	83	215	139	88	12	115	169	58	112
210	18	209	17	79	168	59	148	214	247	235	13	166	232	250	61
104	16	211	123	248	249	233	234	140	25	202	87	243	183	44	239

Non-linearity: The non-linearity NL of a function g can be defined as

$$N(g) = ½(2^{N} - WHT_{max}) \quad (3.1)$$

Where WHT_{max} is the maximum of all the absolute values in the Walsh Hadamard Transform spectrum. The non-linearity of a Boolean function f shows how much the function is different from an affine function having least hamming distance from f. For calculating the non-linearity of the S-box, we can calculate the non-linearity of each of the constituent Boolean functions and take the average of all of them. The fast method of calculating the WHT is used for non-linearity calculation of the chosen S-box and the AES s-box. Since both the s-boxes consist of 8 constituent Boolean functions the non-linearity for all of these is calculated. The results are shown in the following Table 3.2. As we can see the Non-linearity of the chosen s-box is comparable to AES s-box as it is only a little lesser than AES s-box. So, we can say that we can use this chosen s-box for further analysis and can compare it with other s-boxes.

Table 3.3: the non-linearity analysis of S-boxes

S-box	0	1	2	3	4	5	6	7	average
Selected S-box	102	104	98	108	104	102	108	106	104
AES S-box	112	112	112	112	112	112	112	112	112

The time complexity of this implementation is same as fast WHT i.e. O(NlogN) where N is size of input constituent Boolean function.

3.3 PAR value analysis of various s-boxes

PAR is a parameter of quantifying the non-linearity of S-boxes. The PAR value is helpful in linear cryptanalysis of a block cipher. i.e. by calculating the value of PAR we can analyze the non-linearity of an s-box. i.e. the higher the PAR value the easier the approximation. PAR value can be calculated in 2 forms i.e. either as largest PAR value or as a smallest PAR value. The PAR value can be calculated with respect to many transform and it is implemented using WHT, Nega Hadamard Transform and HN Transform set[6]. A system is implemented that takes an s-box as input and calculate its largest and smallest PAR. By comparing the value of PAR we can find out whether it is possible to linearly approximate the function or not. The method of calculating largest and smallest PAR for an S-box is given in the Section 2.12. Here we will analyze the results for DES s-boxes, Serpent s-boxes, AES s-box and the chosen s-box. The results are given in Table 3.4, Table 3.5, Table 3.6 and Table 3.7.

The only difference in calculating the PAR values for the HN Transform set is that in case of largest PAR, the largest value is taken for each matrix and then the largest out of these is taken. But in case of smallest PAR, maximum is taken per matrix and then the smallest of all these is taken as the smallest PAR.

Serpent and DES uses different sizes of S-boxes. So we can't compare their PAR values directly but we can compare them. The largest PAR values of DES S-boxes are quite high as compared to Serpent.

Table 3.4: Smallest and largest PAR for WHT for DES and Serpent 8 s-boxes

S-boxes	Largest PAR(DES)	Largest PAR(Serpent)	Smallest PAR(DES)	Smallest PAR(Serpent)
S-box 0	20.25	4.0	4.0	4.0
S-box 1	16.0	4.0	4.0	4.0
S-box 2	16.0	4.0	6.25	4.0
S-box 3	16.0	4.0	6.25	4.0
S-box 4	25.0	4.0	4.0	4.0
S-box 5	12.25	4.0	4.0	4.0
S-box 6	20.25	4.0	4.0	4.0
S-box 7	16.0	4.0	6.25	4.0

Table 3.5: Smallest and largest PAR for WHT for AES and chosen S-box

	Largest PAR(AES)	Largest PAR (chosen s-box)	Smallest PAR(AES)	Smallest PAR (chosen s-box)
S-box	4.0	18.0625	4.0	5.0625

So, we can say that DES can be more easily approximated using linear expressions as compared to Serpent[7]. Since, higher value of PAR means better linear approximation. By analyzing Table 3.4 we can see that the PAR values of AES S-box are very small meaning that it is highly resistant to linear cryptanalysis. But the PAR values of chosen S-box range from 5 to 18. It means that although it can be used for cryptography but it will vulnerable to linear cryptanalysis as higher the PAR better the Linear Approximation.

As already mentioned a system is also implemented to calculate the largest and smallest PAR of HN Transform also. The system consist of the following steps to calculate the largest PAR:

a. Take input S-box in the form of decimal
b. Covert the s-box into constituent Boolean function and then covert this into a set of Boolean functions consisting of all the linear combinations of the constituent Boolean functions.
c. Convert the input set of Boolean functions into a set of Boolean functions with complex value entries.
d. Use the method previously described to generate HN Transform set and calculate the spectrum for each of the Boolean functions with each of the HN Transform matrix.
e. Now take the maximum absolute value for each of the Boolean function spectrum calculated and take square of this maximum value and multiply it with the input Boolean function size to get the largest PAR. The absolute value is taken by taking the square root of the squares of the real part and the imaginary part of the complex values of the spectrum.

   Similarly the smallest PAR can also be calculated by taking the smallest out of the maximum values taken for each spectrum for each HN Transform matrix.

Table 3.6: Largest and smallest PAR for HN Transform for DES and Serpent

S-boxes	Largest PAR(DES)	Largest PAR(Serpent)	Smallest PAR(DES)	Smallest PAR(Serpent)
S-box 0	20.25	8.0	6.25	4.0
S-box 1	16.0	8.0	6.25	4.0
S-box 2	16.0	8.0	6.625	4.0
S-box 3	18.0	5.0	8.5	4.0
S-box 4	25.0	8.0	5.125	4.0
S-box 5	13.625	8.0	6.25	4.0
S-box 6	20.25	8.0	5.625	4.0
S-box 7	16.0	5.0	6.25	4.0

Table 3.7: Largest PAR for HN Transform for AES and chosen S-box

	Largest PAR(AES)	Largest PAR (chosen s-box)	Smallest PAR(AES)	Smallest PAR (chosen s-box)
S-box	4.0	18.0625	4.0	5.0625

As we can see that PAR values for the HN Transform spectrum are higher than the PAR values of the Walsh Hadamard Transform spectrum for both DES and Serpent, we can say that the approximation is better in case of the HN Transform spectrum than the Walsh Hadamard Transform. This signifies that it may be difficult to approximate the Boolean functions and S-boxes of any cipher system with Z_2 linear expressions but more generalized linear approximations can be easily obtained. Also the new S-box is weaker in case of approximation and can be easily approximated.

3.4 Heuristic Method using PAR

The heuristic method like Hill Climbing[12] are the search methods which start at a specified point and search in the direction of better properties. This method is better than the brute force method which takes in all the points into consideration and doesn't work efficiently for large search spaces. PAR values are another parameter quantifying the non-linearity of s-boxes and Boolean functions. So the heuristic method Hill climbing[12] can make use of it to search better Boolean functions and S-boxes. A new method is proposed which takes concepts of Hill climbing and PAR values of Walsh Hadamard transform and Nega Hadamard Transform to check whether the PAR value inclusion is a better option and also if Nega Hadamard Transform can help in avoiding local maxima conditions. This method is mainly proposed to prove and show the following condition:

"Suppose f has good Nega Hadamard Transform then f $\oplus$ s_2 should have good Walsh Hadamard Transform"

Here s_2 is a homogeneous symmetric Boolean function with 2 variables.

This condition can help us move to better PAR values in the search space and also to avoid local maxima in some cases.

The algorithm proposed can be described as follows:

a. A random function is taken and its PAR values for the Walsh Hadamard Transform and Nega Hadamard Transform is calculated. Two methods PARh and PARn are implemented to calculate these two values.

b. The PAR values calculated are compared and the actions taken are as follows:

   If PAR of WHT is smaller, then hill climbing is used to move to another Boolean function with better PAR value for WHT. i.e. one value in the truth table is changed and the PAR value of the resulting Boolean function is compared.

   If PAR of Nega Hadamard Transform is smaller then 2 variable homogeneous symmetric Boolean function is xored to the Boolean function and the resulting Boolean function is used in step a and the steps are repeated until no better Boolean function is get in the recursion.

   The method for calculating symmetric Boolean function of 2 algebraic degree is also implemented to be used in the above system. The symmetric Boolean function as implemented can also be describe in the following steps:

a. Take the number of variables ‘a’ in the Boolean function.
b. For each of the values from 0 to 2^a, convert the values in binary form and put it in an array.
c. Now take the XOR of all the values obtained by taking and of a value and the value next to it in the Boolean array. This will give the output symmetric Boolean function value.
   For example the following Table 3.8 shows the symmetric Boolean functions in the form of truth table listed horizontally generated from the algorithm.

Table 3.8: Homogeneous symmetric Boolean function of 2 algebraic degree

Number of variables	Homogeneous symmetric Boolean function of degree 2
4	{0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0}
6	{0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1}
8	{0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0,1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1,0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0}

Fig 3.1 shows the flow chart of the above proposed algorithm.

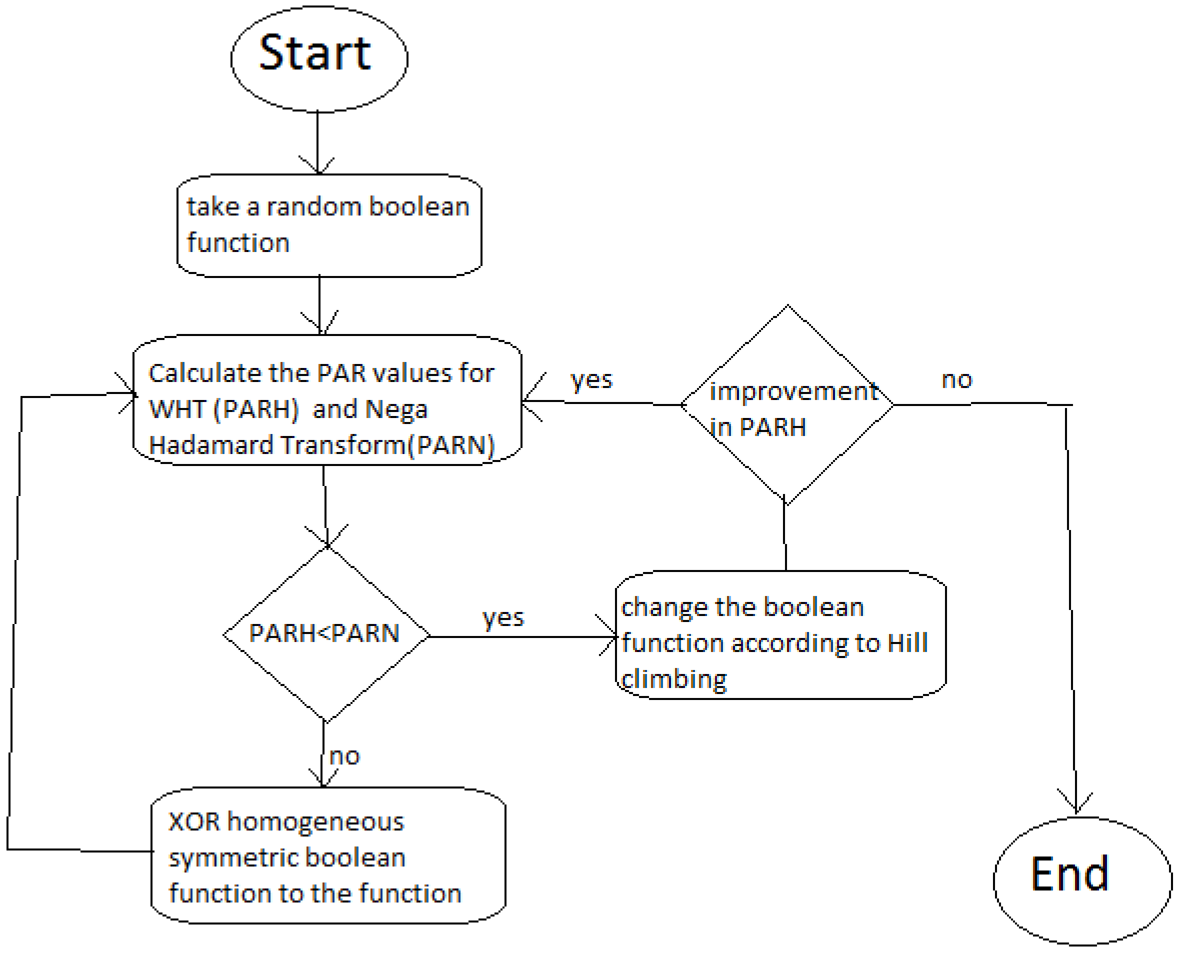
Start
take a random boolean function
Calculate the PAR values for WHT (PARH) and Nega Hadamard Transform(PARN)
yes
improvement in PARH
no
PARH<PARN
yes
change the boolean function according to Hill climbing
no
XOR homogeneous symmetric boolean function to the function
End

Figure 3.1 Flow chart of proposed algorithm

The algorithm shows the expected results. i.e. if the Nega Hadamard transform is better then the new function obtained after XORing homogeneous symmetric Boolean function always gives a Boolean function with better or equal PAR values. This can be shown with the snaps of the algorithm at some points in the search with the help of the following Table 3.9.

Table 3.9: PAR values for 2 variable Boolean function

f 3	f 2	f 1	f 0	PARH	PARN
1	1	1	1	4	1
1	1	1	-1	1	2

Here for 2 variable Boolean function the homogeneous symmetric Boolean function is 1,1,1,-1. As it can be seen in the Table 3.8 since PARH>PARN, so symmetric Boolean function is added and the PAR values improved.

Table 3.10: PAR values for 3 variable Boolean function

F7	F6	F5	F4	F3	F2	F1	F0	PARH	PARN
-1	-1	1	-1	1	1	1	-1	2	1
-1	-1	1	1	1	-1	-1	1	2	1

Here for 3 variable Boolean function the homogeneous symmetric Boolean function is 1 1 1 -1 1 -1 -1 -1. As it can be seen in the Table 3.9 since PARH>PARN, so symmetric Boolean function is added and the PAR values remain same but doesn’t decrease.

The results show that the above algorithm works and the statement is proved to be correct. So a new method is proposed and showed to be working efficiently to find new Boolean functions with better cryptographic properties.

CHAPTER 4: CONCLUSION

Various transforms are useful in analyzing the suitability of Boolean functions and s-boxes for cryptography. An efficient and fast system is implemented for calculating the Walsh Hadamard Transform and Nega Hadamard Transform and the HN transform. The efficient and fast system for Walsh Hadamard Transform was already implemented. But in case of Nega Hadamard Transform a new fast method similar to the Walsh Hadamard Transform was proposed. This system was further extended to calculate the HN Transform too. Since the HN Transfrom set consist of 2^n transform matrices, it takes exponential time to calculate the transform by recursive methods. So, a fast method was developed in which the entry of any of the 2^n matrices can be calculated in constant time. So a useful system for calculating all the Transforms efficiently is developed.

A sample S-box whose non-linearity was comparable to the AES S-box was chosen and the various Transforms and related PAR values are calculated to see if the chosen s-box is suitable for strong cipher systems or not. And the chosen s-box is found to be weaker than the AES S-box.

PAR values are also useful in the sense that they can be used to analyze various s-boxes with respect to linear cryptanalysis.i.e. a system to calculate the PAR values for various transforms including the Walsh Hadamard Transform, the Nega Hadamard Transform and the HN Transform is implemented. Also, higher the PAR value better the linear approximation and more vulnerable to linear cryptanalysis. A system is implemented for calculating the largest and smallest PAR values for all the transforms like WHT, Nega Hadamard Transform and the HN Transform. By using this system various s-boxes are analyzed and compared to each other. A new s-box is also tested against the above implemented system and is found to be weak. By analyzing the results it can be shown that the chosen s-box is weaker than AES s-box against linear cryptanalysis.

Also a new heuristic method is proposed incorporating the hill climbing method which take into consideration the PAR values and it is found to be a good method to find new Boolean functions with better cryptographic properties. Also the fact that “Suppose f has good Nega Hadamard Transform then $f \oplus s_2$ should have good Walsh Hadamard Transform” is also proved to be right. This heuristic method consist of the calculation related to the PAR values with respect to the Walsh Hadamard Transform, the Nega Hadamard Transform and the calculation of symmetric homogeneous Boolean function.

4.1 *Future Work*

We can further extend this work by constructing a more generalized and fast system for Transform calculation which can efficiently calculate various transforms for e.g. WHT, Nega Hadamard Transform, HN transform, HIN transform etc. this system can further be used for calculating non-linearity aspect with respect to these transforms so that linear approximation over generalized set of integers can be done. This will help in looking at the linear cryptanalysis from a different perspective as we will be able to approximate the S-boxes with generalized linear approximations. Other aspects of creating Strong Boolean functions and s-boxes can also be analyzed in future. Also the heuristic method proposed can be further improved to find better Boolean functions with better cryptographic properties. The fact that “Suppose f has good Nega Hadamard Transform then $f \oplus s_2$ should have good Walsh Hadamard Transform” can also be used in a number of ideas since it proved to be correct.

REFERENCES

[1] Burnett, L. D.: Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography. PhD thesis, Queensland University of Technology (2005).

[2] Yngve Ådlandsvik : Generalized Bent and/or Negabent Constructions. thesis (2012).

[3] B. J. Fino and V. R. Algazi, "Unified matrix treatment of the fast Walsh Hadamard transform," IEEE Trans. Compute., vol. C-25, pp. 1142-1146, Nov. 1976.

[4] Hussain, I., Shah, T., Gondal, M.A., Khan, M., Khan, W.A.: Construction of new S-box using a linear fractional transformation. World Appl. Sci. J. **14**(12), 1779–1785 (2011).

[5] S. Mister and C. Adams, "Practical S-box design", *Workshop Record Selected Areas Cryptography (SAC',96)*, pp.61 -76 1996.

[6] M. G. Parker: *Generalized S-Box Nonlinearity NESSIE Public Document*, 2003.

[7] S. Gangopadhyay, E. Pasalic, and P. Stˇanicˇa, "A note on generalized bent criteria for Boolean functions," IEEE Trans. Inf. Theory, vol. 59, no. 5, pp. 3233–3236, May 2013.

[8] M.Matsui, "Linear cryptanalysis method for DES Cipher," *Advances in Cryptology-EUROCRYPT '93,* LNCS 765, pp.386-397 , 1994.

[9] F.J. Macwilliams and N.J. Sloane, *Theory of Error-Correcting Codes,*North Holland, Amsterdam, 1978.

[10] T. W. Cusick and P. Stănică, Cryptographic Boolean Functions and Applications, Academic Press, San Diego, CA, 2009.

[11] W. Millan, A. Clark, and E. Dawson. Smart Hill Climbing Finds Better Boolean Functions. In Workshop on Selected Areas in Cryptology 1997, Workshop Record, pages 50-63, 1997.

[12] Millan,W., Clark, A. and Dawson, E., "Boolean Function Design Using Hill Climbing Methods," in 4th Australian Conference on Information Security and Privacy (Schneier, B. ed), LNCS, 1587, pp. 1-11, Springer-Verlag, April 1999.

[13] Ross Anderson, Eli Biham and Lars Knudsen: *Serpent: A Proposal for the Advanced Encryption Standard*.

APPENDIX A. THE S-BOXES USED IN THE THESIS WORK:

1. DES S-BOXES IN DECIMAL FORM

Table A.1: S_1 S-Box of DES

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Table A.2: S_2 S-Box of DES

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Table A.3: S_3 S-Box of DES

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Table A.4: S_4 S-Box of DES

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Table A.5: S_5 S-Box of DES

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Table A.6: S_6 S-Box of DES

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Table A.7: S_7 S-box of DES

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Table A.8: S_8 S-Box of DES

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

2. SERPENT S-BOXES IN DECIMAL FORM

Table A.9: Serpent S-boxes in decimal form

S0	3	8	15	1	10	6	5	11	14	13	4	2	7	0	9	12
S1	15	12	2	7	9	0	5	10	1	11	14	8	6	13	3	4
S2	8	6	7	9	3	12	10	15	13	1	14	4	0	11	5	2
S3	0	15	11	8	12	9	6	3	13	1	2	4	10	7	5	14
S4	1	15	8	3	12	0	11	6	2	5	4	10	9	14	7	13
S5	15	5	2	11	4	10	9	12	0	3	14	8	13	6	7	1
S6	7	2	12	5	8	4	6	11	14	9	1	15	13	3	10	0
S7	1	13	15	0	14	8	2	11	7	4	12	10	9	3	5	6

3. AES S-BOX IN HEXADECIMAL FORM

Table A.10: AES S-box in Hexadecimal form

63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB

E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16